ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΙΧΝΕΥΣΗ ΚΑΤΑΣΤΑΣΗΣ ΟΔΗΓΟΥ ΩΣ ΠΡΟΣ ΤΗΝ ΙΚΑΝΟΤΗΤΑ ΟΔΗΓΗΣΗΣ ΤΟΥ ΜΕ ΤΗ ΧΡΗΣΗ IoT (ΥΠΝΗΛΙΑ, ΑΝΗΣΥΧΗ ΣΥΜΠΕΡΙΦΟΡΑ, ΝΕΥΡΙΚΟΤΗΤΑ)

Χρήστος Χούθης

ΑΜ: 161094

Επιβλέπων Καθηγητής: Ιωάννης Βογιατζής

Αθήνα

Σεπτέμβριος 2021

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΙΧΝΕΥΣΗ ΚΑΤΑΣΤΑΣΗΣ ΟΔΗΓΟΥ ΩΣ ΠΡΟΣ ΤΗΝ ΙΚΑΝΟΤΗΤΑ ΟΔΗΓΗΣΗΣ ΤΟΥ ΜΕ ΤΗ ΧΡΗΣΗ IoT (ΥΠΝΗΛΙΑ, ΑΝΗΣΥΧΗ ΣΥΜΠΕΡΙΦΟΡΑ, ΝΕΥΡΙΚΟΤΗΤΑ)

Μέλη Εξεταστικής Επιτροπής:

1. Βογιατζής Ιωάννης
2. Γιαννακόπουλος Παναγιώτης
3. Φατούρος Σταύρος

Η πτυχιακή/διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	**Επώνυμο-Όνομα**	**Ιδιότητα**	**Ψηφιακή υπογραφή**
1	Βογιατζής Ιωάννης	Καθηγητής	Ioannis Voyiatzis Digitally signed by Ioannis Voyiatzis Date: 2022.06.07 14:43:11 +03'00'
2	Γιαννακόπουλος Παναγιώτης	Καθηγητής	Panagiotis Giannakopoulos Digitally signed by Panagiotis Giannakopoulos Date: 2022.06.07 23:04:14 +03'00'
3	Φατούρος Σταύρος	Αναπληρωτής Καθηγητής	Stavros Fatouros Digitally signed by Stavros Fatouros Date: 2022.06.07 15:05:03 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **ΧΟΥΘΗΣ ΧΡΗΣΤΟΣ** του **ΕΛΕΥΘΕΡΙΟΥ**, με αριθμό μητρώου **711161094** φοιτητής του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο/Η Δηλών/ούσα

Χούθης Χρήστος

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω, τον επιβλέποντα καθηγητή μου κ. Ιωάννη Βογιατζή, για την εμπιστοσύνη που μου έδειξε, από την αρχή που ανέλαβα τη παρούσα διπλωματική εργασία, καθώς την υπομονή, τη στήριξη και τις συμβουλές που μου παρείχε κατά την διάρκεια εκπόνησης της εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου, όπου με τον τρόπο τους με στηρίξαν και μου συμπαρασταθήκαν ψυχολογικά, κατά τη διάρκεια, εκπόνησης την εργασίας μου.

Περίληψη

ΑΝΙΧΝΕΥΣΗ ΚΑΤΑΣΤΑΣΗΣ ΟΔΗΓΟΥ ΩΣ ΠΡΟΣ ΤΗΝ ΙΚΑΝΟΤΗΤΑ ΟΔΗΓΗΣΗΣ ΤΟΥ ΜΕ ΤΗ ΧΡΗΣΗ IoT (ΥΠΝΗΛΙΑ, ΑΝΗΣΥΧΗ ΣΥΜΠΕΡΙΦΟΡΑ, ΝΕΥΡΙΚΟΤΗΤΑ)

Η μάστιγα όλων των εποχών ήταν και είναι τα τροχαία ατυχήματα. Σε μεγάλο ποσοστό αυτά συμβαίνουν λόγο του ότι ο οδηγός δεν είναι πάντα σε θέση να οδηγήσει, λόγο κούρασης, άγχους, υπνηλίας, περίεργης συμπεριφοράς, ομιλίας στο κινητό κ.α. Σκοπός αυτής της διπλωματικής εργασίας είναι η αποδοτική ανάπτυξη λογισμικού στην πλατφόρμα Raspberry Pi, χρησιμοποιώντας κάμερα υπερύθρων με LDR Sensors, IR LED και μηχανική όραση για την ανίχνευση της κατάστασης του οδηγού, με σκοπό την λήψη απόφασης εάν ο οδηγός είναι σε θέση να οδηγήσει το όχημα με ασφάλεια. Ανάλογα με την κατάσταση του οδηγού το σύστημα θα αναπαράγει φωνητικά μηνύματα συμβουλευτικής φύσεως, και εάν ο οδηγός δεν είναι σε θέση να οδηγήσει το όχημα μετά ένα όριο ορισμένων προειδοποιήσεων τότε μέσω IoT και με χρήση Wi-Fi, GSM/GPRS-GPS module, θα ενημερώνει τα οικεία πρόσωπα του οδηγού μέσω ηλεκτρονικού μηνύματος για την κατάσταση και την τοποθεσία όπου αυτός βρίσκεται.

Λέξεις-Κλειδιά

Raspberry Pi, OpenCV, GSM/GPRS-GPS, LDR Sensors, Μηχανική όραση, Ανίχνευση προσώπου, Κάμερα υπερύθρων, Διαδίκτυο των πραγμάτων

Abstract

DRIVER STATUS DETECTION FOR CAPABILITY OF DRIVING USING IoT (DROWSINESS, ANXIOUS BEHAVIOUR, IRRITATED BEHAVIOUR)

The scourge of all time was and are the car accidents. A large percentage of these occur due to the fact that the driver is not always able to drive, due to fatigue, anxiety, drowsiness, strange behavior, talking on the cell phone, etc. The purpose of this thesis is the efficient development of software on the Raspberry Pi platform, using an infrared camera with LDR sensors, IR LED and mechanical vision to detect the driver's condition, in order to decide if the driver is able to drive the vehicle with safety. Depending on the driver's condition the system will play advisory voice messages, and if the driver will be unable to drive the vehicle after a certain warning limit then via IoT and using Wi-Fi, GSM/GPRS-GPS module, it will inform his relatives via email for his status and his location.

Keywords

Raspberry Pi, OpenCV, GSM/GPRS-GPS, LDR Sensors, Machine Vision, Face Detection, Infrared Camera, Internet of Things

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

IoT	Internet of Things
LDR	Light Dependent Resistor
IR	Infrared Radiation
GSM	Global System for Mobile Communications
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPIO	General Purpose input/output
LED	Light Emitting Diode
CPU	Central Processing Unit
GPU	Graphics Processing Unit
FPS	Frames Per Second
UML	Unified Modeling Language
APN	Access Point Name
UART	Universal Asynchronous Receiver-Transmitter
TN	Τεχνητή Νοημοσύνη

1. Εισαγωγή

1.1 Γενικά

Η ανάγκη για την καθημερινή μετακίνηση καθώς και η ανάγκη για την διάνυση μεγάλων αποστάσεων και τη μεταφορά φορτιού, οδήγησαν τον άνθρωπο σε αναζήτηση διαφόρων μέσων μεταφοράς, τα οποία θα μπορούσαν να εξυπηρετούν τις καθημερινές του ανάγκες.

Με αυτόν τον τρόπο η τεχνολογική εξέλιξη των μεταφορικών μέσων και ιδίως του αυτοκινήτου ως μεταφορικό μέσο, άρχισε να τίθεται σε παγκόσμια χρήση και να αναπτύσσεται κατά τον $20^{ο}$ αιώνα. Η ραγδαία τεχνολογική ανάπτυξη στον $21^{ο}$ αιώνα και η συντριπτική πλειοψηφία των μεταφορικών μέσων να είναι πλέον τα αυτοκίνητα, αποτέλεσε τον παράγοντα ώστε σήμερα, ο αριθμός των αυτοκινήτων να ανέρχεται περίπου στα 1.4 δισεκατομμύρια παγκοσμίως.

Εκτός από τα κοινωνικά και οικονομικά οφέλη που επέφερε το αυτοκίνητο στον άνθρωπο, προκάλεσε και πολλά προβλήματα, με ένα από τα μεγαλύτερα να είναι τα τροχαία ατυχήματα τα οποία όλο και συνεχώς αυξάνονται. Σε μεγάλο ποσοστό αυτά συμβαίνουν διότι ο οδηγός δεν είναι πάντα σε θέση να οδηγήσει, λόγο κούρασης, άγχους, υπνηλίας, περίεργης συμπεριφοράς ή ομιλίας στο κινητό.

1.2 Αντικείμενο Διπλωματικής Εργασίας

Όπως αναφέραμε ένας από του μεγαλύτερους λόγους, που μπορεί να πραγματοποιηθεί ένα τροχαίο ατύχημα, αφορά την κατάσταση του οδηγού και σε τι θέση βρίσκεται ώστε να μπορέσει να αντιδράσει. Σε περίπτωση που ο οδηγός δεν είναι σε θέση να οδηγήσει λόγο κούρασης, άγχους, υπνηλίας, περίεργης συμπεριφοράς ή ομιλίας στο κινητό, συνεπώς θέτει τον εαυτό του αλλά και τους υπόλοιπους σε κίνδυνο.

Η παρούσα διπλωματική λοιπόν, δίνει έμφαση στα προαναφερόμενα προβλήματα και χρησιμοποιώντας τις κατάλληλες τεχνικές, προσπαθεί να παρακολουθεί τις κινήσεις του οδηγού, ώστε να είναι σε θέση να εντοπίζει με όσο τον δυνατόν καλύτερο τρόπο, την συμπεριφορά του, κατά την διάρκεια της οδήγησης.

Τα χαρακτηριστικά του οδηγού που παρακολουθούνται είναι τα μάτια για πιθανόν υπνηλία, το στόμα για πιθανόν χασμουρητό και τα χαρακτηριστικά του σώματος του. Από τα χαρακτηριστικά σημεία του σώματος του παρακολουθούνται, τα χέρια, τα οποία μπορεί πιθανόν να βρίσκονται σε μη κανονική θέση ή ο οδηγός να μιλάει στο κινητό. Επίσης, ένα ακόμα χαρακτηριστικό που παρακολουθείται, είναι η πιθανόν κλίση του κεφαλιού προς τον αριστερό ή δεξιό ώμο αντίστοιχα, όπου μπορεί πιθανόν να μας υποδείξει πως ο οδηγός είναι αρκετά κουρασμένος.

Έπειτα από την διαδικασία εντοπισμού, ένας συναγερμός ενεργοποιείται και ένα μήνυμα συμβουλευτικής φύσεως αναπαράγεται στον οδηγό, με σκοπό την συγκέντρωση και την εστίαση προσοχής του. Σε περίπτωση επανειλημμένων εντοπισμών, το GSM/GPRS Module ενεργοποιείται για την παροχή πρόσβασης στο διαδίκτυο, το GPS Module εντοπίζει την τοποθεσία του οδηγού, και με την διαμόρφωση κατάλληλου μηνύματος, η τοποθεσία στέλνεται ηλεκτρονικά μέσω ηλεκτρονικού μηνύματος στο email του παραλήπτη που έχει καθοριστεί. Πλέον ο παραλήπτης στο μήνυμα του, μπορεί να δει στοιχεία της τοποθεσίας του οδηγού, καθώς και την ακριβή του τοποθεσία του.

1.3 Διάρθρωση Διπλωματικής

Στο 2^{ο} κεφάλαιο υπάρχει μια εισαγωγή στην Τεχνητή Νοημοσύνη, καθώς και στα επιστημονικά πεδία που την αποτελούν. Βλέπουμε θεωρητικές έννοιες και στοιχεία τα οποία θα αναφέρουμε και μέσα στην εργασία μας.

Στο 3^{ο} κεφάλαιο αναφερόμαστε σε μοντέλα και μεθόδους ανίχνευσης προσώπου, ανίχνευσης χαρακτηριστικών προσώπου και ανίχνευσης χαρακτηριστικών σώματος. Ακόμα, βλέπουμε τον τρόπο λειτουργίας των μέθοδών αυτών, καθώς και τα πλεονεκτήματα και μειονεκτήματα τους.

Στο 4^{ο} κεφάλαιο αναλύουμε και περιγράφουμε πλήρως το υλικό που χρησιμοποιήθηκε για την κατασκευή του συστήματος μας. Επίσης, βλέπουμε όλη την διαδικασία συναρμολόγησης του υλικού βήμα-βήμα.

Στο 5^{ο} κεφάλαιο περιγράφουμε το λογισμικό πάνω στο οποίο δημιουργήσαμε το σύστημα μας, καθώς και τις απαραίτητες βιβλιοθήκες που χρησιμοποιήθηκαν για την κατασκευή του συστήματος.

Στο 6^{ο} κεφάλαιο παρουσιάζουμε βήμα-βήμα όλη την απαραίτητη διαδικασία εγκατάστασης του λογισμικού, των βιβλιοθηκών, των πακέτων αλλά και όλων των απαραίτητων τροποποιήσεων, ώστε να θέσουμε το σύστημα μας σε λειτουργία.

Στο 7^{ο} κεφάλαιο βλέπουμε το διάγραμμα δραστηριότητας στον οποίο βασίζεται η διαδικασία εκτέλεσης του συστήματος μας , αναλύουμε τις λειτουργίες του έργου μας, βλέπουμε τις διαφορές ανάμεσα σε δύο διαφορετικά μοντέλα ανίχνευσης προσώπου και τέλος θέτουμε το σύστημα μας σε λειτουργία βλέποντας και σχολιάζοντας κάποια πραγματικά αποτελέσματα.

Στο 8^{ο} κεφάλαιο βλέπουμε τα συμπεράσματα και τις μελλοντικές προτάσεις που αφορούν την βελτίωση του συστήματος μας.

2. Τεχνητή Νοημοσύνη

2.1 Εισαγωγή

Η Τεχνητή Νοημοσύνη (Artificial Intelligence - AI) αποτελεί έναν κλάδο της επιστήμης των υπολογιστών, ο οποίος τυπικά διατυπώθηκε σαν όρος το 1956 στη συνάντηση μερικών επιφανών επιστημόνων, όπως ο John McCarthy, ο Marvin Minsky, ο Claude Shannon κ.α. Στην πραγματικότητα φαίνεται να είχε ήδη εμφανιστεί από το 1950, σε μελέτη του Alan Turing, ο οποίος είχε θέσει το ερώτημα <<Μπορούν οι μηχανές να σκεφτούν;>>. Βεβαίως το ερώτημα αυτό δεν μπορούσε να απαντηθεί αλλά αποτέλεσε την αρχή ώστε ο Alan Turing να προτείνει την δοκιμή Turing, η οποία αποτελούσε μια απλή δοκιμή όπου θα μπορούσε να αποδείξει αν μια μηχανή μπορεί αν διαθέτει ευφυία.

2.2 Ορισμός Τεχνητής Νοημοσύνης

Η ΤΝ αποτελεί τομέα της επιστήμης των υπολογιστών, που ασχολείται με την σχεδίαση νοημόνων υπολογιστικών συστημάτων, που επιδεικνύουν χαρακτηριστικά τα οποία συστήματα μπορούν να:

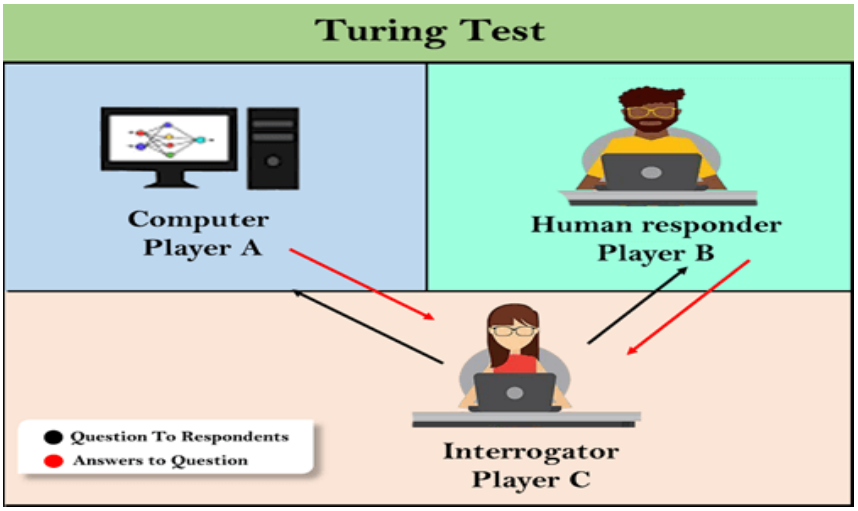
- Σκέφτονται όπως οι άνθρωποι
- Να συμπεριφέρονται όπως οι άνθρωποι
- Σκέφτονται ορθολογικά
- Αντιδρούν ορθολογικά
- Μαθαίνουν από εμπειρίες και παραδείγματα

Ο John Haugeland (1985) αναφέρει πως η ΤΧ νοημοσύνη είναι “Η συναρπαστική νέα προσπάθεια για να κάνουμε τους υπολογιστές να σκέπτονται … μηχανές με νόηση, με την πλήρη και κυριολεκτική έννοια.” [1]

Τα συστήματα τεχνητής νοημοσύνης είναι δυνατόν, να προσαρμόζουν την συμπεριφορά τους, μέχρι ένα βαθμό, αναλύοντας τις πράξεις των προηγούμενων δράσεων τους και επιλύοντας προβλήματα με αυτονομία. [2]

2.3 Δοκιμασία Turing και Σημαντικά Πεδία της ΤΝ

Η δοκιμασία Turing, προτάθηκε από τον μαθηματικό Alan Turing (1950), με σκοπό να παρέχει έναν ικανοποιητικό επιχειρησιακό ορισμό της νοημοσύνης. Στην εν λόγο δοκιμασία ο Turing πρότεινε μια παραλλαγή ενός παιχνιδιού που ονομάζεται <<παιχνίδι της μίμησης>> το οποίο παίζεται από τρεις παίχτες, άγνωστους μεταξύ τους. Οι δύο είναι <<μάρτυρες>> και αντίθετου φύλλου και ο τρίτος είναι ο <<ανακριτής>>, ο οποίος προσπαθεί να διαχωρίσει την ταυτότητα του κάθε μάρτυρα με βάση των απαντήσεων τους σε κάποιες ερωτήσεις που τους θέτονται. Η διαφορά με την δοκιμασία του Turing, είναι ότι αντί της θέσης του ενός <<μάρτυρα>> εισήγαγε μια μηχανή. Επομένως ο <<ανακριτής>>, έπρεπε πλέον όχι μόνο να διακρίνει τα φύλλα των <<μαρτύρων>> αλλά και να διακρίνει τον άνθρωπο από την μηχανή. Αν λοιπόν ο <<ανακριτής>> έκανε λάθος σε αυτή την διάκριση τότε η μηχανή θα μπορούσε να περάσει την δοκιμασία (Εικόνα 2.1). [3].



Εικόνα 2.1 Απεικόνιση της δοκιμασίας Turing **[4]**

Η πρόβλεψη του Turing μπορεί να μην επαληθεύτηκε, καθώς για να περάσει ένας υπολογιστής μια τέτοια δοκιμασία, θα πρέπει να έχεις τις εξής ικανότητες:

- **Επεξεργασία φυσικής γλώσσας**, ώστε να μπορεί να επικοινωνεί ικανοποιητικά
- **Αναπαράσταση γνώσης,** ώστε να αποθηκεύει αυτά που ήδη γνωρίζει και ακούει
- **Αυτοματοποιημένη συλλογιστική,** ώστε να μπορεί να χρησιμοποιεί της αποθηκεμένες πληροφορείς

- **Μηχανική μάθηση,** ώστε να προσαρμόζεται σε νέες καταστάσεις και να εντοπίζει νέα πρότυπα

Η δοκιμασία Turing απέφυγε την άμεση φυσική αλληλεπίδραση μεταξύ του <<ανακριτή>> και του υπολογιστή, διότι η φυσική προσομοίωση του ανθρώπου δεν είναι απαραίτητη για τη νοημοσύνη. Εντούτοις, η πλήρης δοκιμασία Turing περιλαμβάνει και οπτική σήμα, πράγμα που σημαίνει ότι ο υπολογιστής θα χρειαστεί επιπλέον χαρακτηριστικά όπως:

- **Μηχανική όραση**, ώστε να αντιλαμβάνεται αντικείμενα
- **Ρομποτική,** ώστε να έχει δυνατότητα κίνησης και χειρισμό αντικειμένων

Έτσι με αυτόν τον τρόπο, περιγράψαμε συνοπτικά έξι βασικά επιστημονικά πεδία της Τεχνητής Νοημοσύνης. [1]

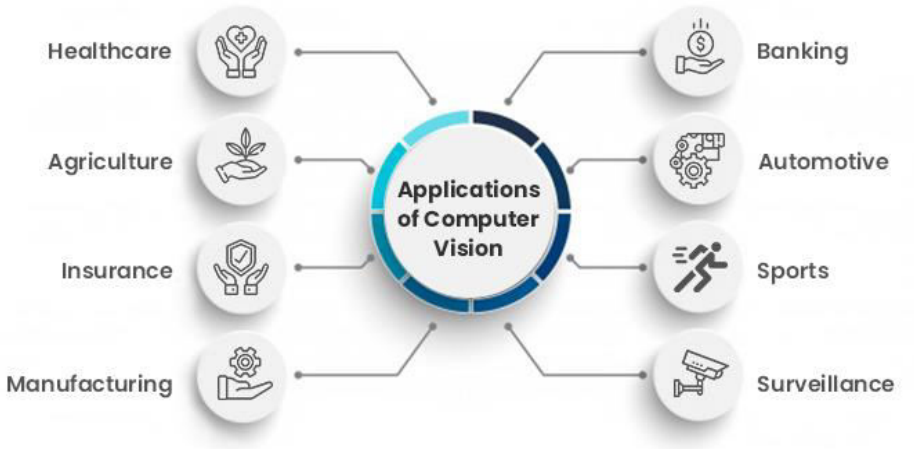
2.4 Μηχανική Όραση

Η Μηχανική Όραση (Machine Vision), Υπολογιστική Όραση (Computer Vision) ή αλλιώς Τεχνητή Όραση (Artificial Vision) αποτελεί ένα επιστημονικό πεδίο της τεχνητής νοημοσύνης, που επιχειρεί να αναπαράγει αλγοριθμικά την αίσθηση της όρασης, μέσω ηλεκτρονικού υπολογιστή ή ρομπότ. Η μηχανική όραση σχετίζεται άμεσα, με την τεχνολογία της σχεδίασης και κατασκευής συστημάτων, που λαμβάνουν και αναλύουν ψηφιακές εικόνες. [5]

Στόχος της τεχνητής όρασης αποτελεί η εξαγωγή χρήσιμων πληροφοριών από ψηφιακές εικόνες, με πρόβλημα την πολυπλοκότητα και τον όγκο των οπτικών δεδομένων προς επεξεργασία σε μια εικόνα. Βέβαια τη λύση του προβλήματος, δίνουν τα τελευταίας τεχνολογίας υπολογιστικά συστήματα, τα οποία διαθέτουν μεγάλη επεξεργαστική ισχύ, ισχυρές κάρτες γραφικών, περισσότερη μνήμη και αποθηκευτικό χώρο. Δεν είναι όμως πάντα έτσι, καθώς υπάρχουν και υπολογιστικά συστήματα (π.χ. Raspberry Pi κτλ.), όπου δίνουν το πλεονέκτημα της εύκολης μεταφοράς και του μικρού μεγέθους τους, αλλά τα χαρακτηριστικά τους δεν δύνανται να είναι, τόσο <<ισχυρά>> όσο ενός <<ισχυρού>> υπολογιστικού συστήματος. Επομένως, η λύση βρίσκεται στην επιλογή μοντέλων και αλγορίθμων, οι οποίοι απαιτούν την ελάχιστη δυνατή πολυπλοκότητα, πάντα με το καλύτερο δυνατό αποτέλεσμα.

Η τεχνητή όραση βρίσκει εφαρμογή σε διάφορους τομείς, με μερικούς από αυτούς να είναι:

- **Η Υγεία,** όπου τα περισσότερα ιατρικά δεδομένα βασίζονται σε εικόνες, και δίνουν αρκετές πληροφορίες για διάγνωση του ασθενή.
- **Η Γεωργία,** όπου βοηθά στην εκτέλεση γεωργικών δραστηριοτήτων.
- **Η Βιομηχανία,** καθώς χρησιμοποιείται για ποιοτικό έλεγχο τελικών προϊόντων.
- **Τράπεζες,** ελέγχοντας πλαστά νομίσματα και απάτες.
- **Αθλητισμό,** προσφέροντας αυτοματοποιημένη αθλητική παραγωγή, παρακολούθηση αθλητών βάση των κινήσεων τους κτλ.
- **Επιτήρηση,** σε δημόσιους χώρους και μη, για λόγους ασφαλείας με μεθόδους όπως αναγνώριση προσώπου, ανίχνευση πλήθους, ανίχνευση παράνομης στάθμευσης κτλ.
- **Αυτοκινητοβιομηχανία,** όπου πλέον είναι δυνατόν η ανάπτυξη αυτοκινήτων αυτόνομης οδήγησης (όπως Tesla), όπου όχι μόνο προσφέρουν αυτόνομη οδήγηση, αλλά ίσως στην πορεία μειώσουν τα ατυχήματα, καθώς ελαχιστοποιούν τα ανθρώπινα σφάλματα όσο εξελίσσονται. [6]



Εικόνα 2.2 8 τομείς εφαρμογής της μηχανικής όρασης **[6]**

2.5 Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning) αποτελεί υποπεδίο της τεχνητής νοημοσύνης το οποίο, δίνει την δυνατότητα δημιουργίας μηχανών ικανών να μαθαίνουν, να βελτιώνουν την απόδοση τους βάση των υπαρχόντων γνώσεων τους, με σκοπό την δημιουργία μοντέλων ή προτύπων από σύνολα δεδομένων. [7]

Η Μηχανική Μάθηση συνδέεται άμεσα με τον τομέα των μαθηματικών και συγκεκριμένα με τους κλάδους της, όπως η θεωρία πιθανοτήτων, η γραμμική άλγεβρα κτλ. Η χρήση των μαθηματικών αποτελεί απαραίτητο εργαλείο για την βελτίωση και την επιλογή αλγορίθμων με την συνεκτίμηση πολλών παραμέτρων (πολυπλοκότητα μοντέλου, ακρίβεια πρόβλεψης, χρόνος εκπαίδευσης αλγορίθμου κτλ.).

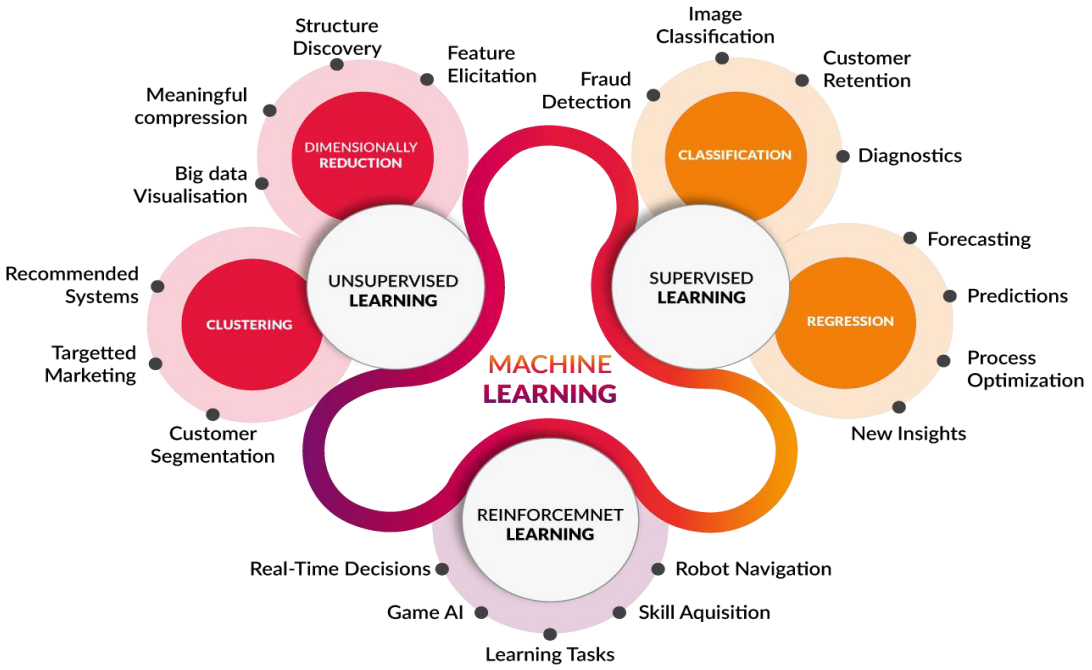
Ένα από τα βασικά στοιχεία της μηχανική μάθησης είναι η δημιουργία μοντέλων και αλγορίθμων οι οποίοι θα μπορούν να αναγνωρίζουν πολυσύνθετα σχέδια και θα λαμβάνουν νοήμονες αποφάσεις με βάση τα δεδομένα της εισόδου που λαμβάνουν. Το κυρίως πρόβλημα είναι ότι τα δεδομένα εισόδου, μπορεί να είναι πολύ μεγάλα έτσι ώστε να μπορέσουμε να τα συμπεριλάβουμε όλα στην εκπαίδευση του μοντέλου. Έτσι για να πραγματοποιηθεί η εκπαίδευση του μοντέλου θα πρέπει να τα δεδομένα εισόδου να γενικευτούν έτσι ώστε να μπορέσουν να παραχθούν χρήσιμα συμπεράσματα για νέα προβλήματα.

2.5.1 Είδη Μηχανικής Μάθησης

Ο τομέας την μηχανικής μάθησης χωρίζεται σε τρεις κατηγορίες μάθησης, όπου η χρήση της εκάστοτε κατηγορίας, βασίζεται στην φύση του προβλήματος που θέλουμε να επιλύσουμε. Παρακάτω βλέπουμε τις τρεις κατηγορίες:

- **Επιβλεπόμενη Μάθηση (Supervised Learning):** Το σύστημα καλείται να μάθει την περιγραφή του μοντέλου από ένα σύνολο δεδομένων, τα οποία αποτελούν παραδείγματα εισόδου-εξόδου. Με την έννοια της επίβλεψης εννοούμε πως υπάρχει ο επιβλέπων ο οποίος δίνει την σωστή τιμή εξόδου του συστήματός για κάθε σύνολο δεδομένων.
- **Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning):** Το σύστημα θα πρέπει να ανακαλύψει την δομή των δεδομένων εισόδου και στην συνέχεια να κατασκευάσει μοντέλα συσχέτισης για ένα σύνολο δεδομένων χωρίς όμως να γνωρίζει τις επιθυμητές εξόδους. Σε αυτήν την περίπτωση δηλαδή δεν υπάρχει επιβλέπων ο οποίος να δίνει σωστά αποτελέσματα εξόδου, με το σύστημα είναι υπεύθυνο για αυτό.
- **Ενισχυτική Μάθηση (Reinforcement Learning):** Το σύστημα εκπαιδεύεται μέσω της αλληλεπίδρασης του με το περιβάλλον, μέσω δοκιμών και

σφαλμάτων, λαμβάνοντας ως ανατροφοδότηση τις δίκες του εμπειρίες και ενέργειες.

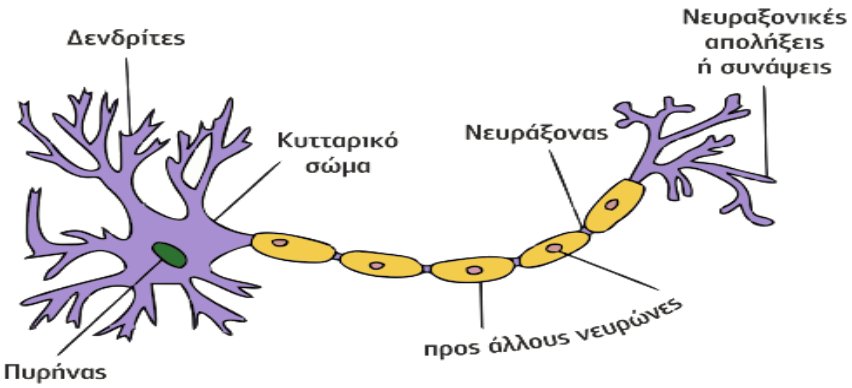


Εικόνα 2.3 Τα 3 είδη μηχανικής μάθησης με τις εφαρμογές τους **[8]**

2.6 Νευρωνικά Δίκτυα

Τα Νευρωνικά Δίκτυα (Neural Networks) αποτελούν ένα σύνολό αλγορίθμων οι οποίοι προσπαθούν να μιμηθούν την συμπεριφορά των βιολογικών νευρώνων ενός ανθρώπινου εγκεφάλου. Αποτελούν μια από τις γνωστές τεχνικές για προβλήματα που αφορούν την αναγνώριση προτύπων (Pattern Recognition) συμπεριλαμβανομένου και της τεχνικής για την ανίχνευση προσώπου (Face Detection).

Προτού δούμε τον τρόπο λειτουργίας των τεχνητών νευρωνικών δικτυών, ας δούμε την λειτουργία του βιολογικού νευρώνα. Η βασική λειτουργία του βιολογικού νευρώνα είναι η μεταφορά σημάτων στους υπόλοιπους νευρώνες μέσω του νευρωνικού δικτύου. Οι νευρώνες αποτελούνται από 3 βασικά στοιχεία, το σώμα κυττάρου, τους νευροάξονες και τους δενδρίτες (Εικόνα 2.3).



Εικόνα 2.3 Διάγραμμα ενός βιολογικού νευρώνα

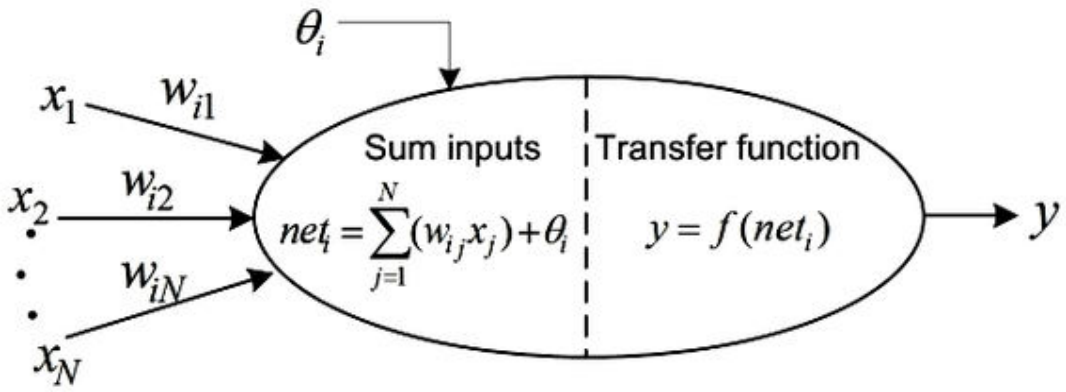
Οι νευροάξονες ενός κυττάρου, συνδέονται με τους δενδρίτες του άλλου μέσω των συνάψεων. Όταν ένας νευρώνας ενεργοποιείται, πυροδοτεί ένα σήμα κατά μήκος του νευροάξονα, το οποίο διασχίζει την σύναψη προς το επόμενο κύτταρο. Ένας νευρώνας ενεργοποιείται μόνο όταν το σήμα που φτάσει από τούς δενδρίτες ξεπερνάει το όριο της πυροδότησης.

Ένας τεχνητός νευρώνας προσπαθεί να προσομοιώσει την λειτουργία του βιολογικού νευρώνα, και αυτό που κάνει είναι να δέχεται πολλές εισόδους x_i, όπου η κάθε είσοδος x_i αποτελείται και από έναν συντελεστή βαρύτητας w_i, και τέλος ο νευρώνας έχει μια μοναδική έξοδο y. Το άθροισμα των δεδομένων εισόδου με τα αντίστοιχα βάρη, υπολογίζεται μέσω της συνάρτησης αθροίσματος:

$$\mathrm{F} = \sum_{i}^{n} x_i \, w_i$$

Έπειτα ο νευρώνας με βάση την συνάρτηση μεταφοράς-μετάβασης, δίνει τιμή εξόδου εφόσον το άθροισμα των εισόδων του, είναι μεγαλύτερο από την τιμή κατωφλίου θ. Ο τύπος της συνάρτηση μεταφοράς, δίνεται από τον παρακάτω τύπο:

$$\mathrm{F} = \sum_{i}^{n} x_i \, w_i - \theta > 0$$



Εικόνα 2.4 Γραφική αναπαράσταση λειτουργίας νευρώνα **[9]**

3. Ανίχνευση Προσώπου & Χαρακτηριστικών Σώματος-Πόζας

3.1 Εισαγωγή

Η ανίχνευση αντικειμένου (object detection), χρησιμοποιείται στην μηχανική όραση για την ανίχνευση αντικειμένων σε μια εικόνα ή ένα βίντεο, και τα οποία αντικείμενα ανήκουν σε μια συγκεκριμένη κατηγορία (όπως πρόσωπα, αυτοκίνητα, κτήρια κτλ.). Η ανίχνευση προσώπου, αποτελεί μια κατηγορία από αυτές η οποία έχει κατά καιρούς απασχολήσει τον τομέα της μηχανικής όρασης, και ως τώρα έχουν προταθεί αρκετά αξιόπιστα μοντέλα εντοπισμού του.

Η ανίχνευση σώματος-πόζας ενός ανθρώπου χρησιμοποιείται στην μηχανική όραση για τον εντοπισμό και την παρακολούθηση των χαρακτηριστικών του σώματος του ανθρώπου (όπως αυτιά, ώμοι , χέρια, πόδια κτλ.). Η ανίχνευση σώματος-πόζας μας δίνει την δυνατότητα χρήσης της σε ένα μεγάλο εύρος τομέων, κάποιους από τους οποίους μπορεί να είναι η ρομποτική, η επαυξημένη πραγματικότητα (virtual reality), τα παιχνίδια κτλ.

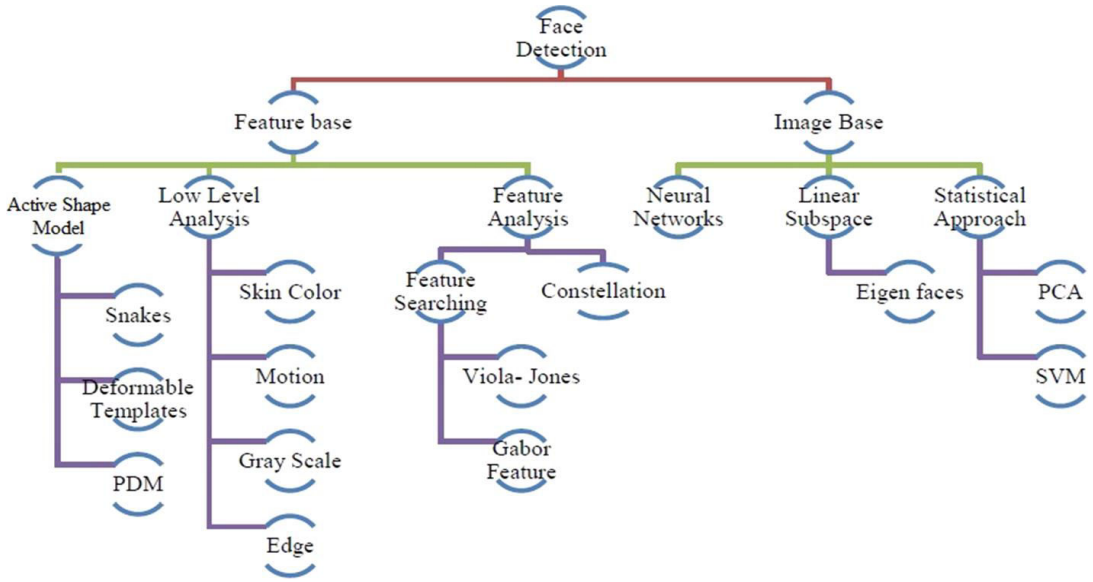
Σε αυτό το κεφάλαιο λοιπόν, θα αναφερθούμε αρχικά σε τεχνικές που χρησιμοποιούνται για την ανίχνευση προσώπου, αναφέροντας τα πλεονεκτήματα και τα μειονεκτήματα της κάθε τεχνικής με βάση την ταχύτητα, την αξιοπιστία, την δυνατότητα τους να τρέχουν σε πραγματικό χρόνο, κτλ. Έπειτα θα δούμε το προ εκπαιδευμένο μοντέλο dlib για την εκτίμηση 68 σημείων πάνω στο πρόσωπο, με βάση τα οποία μπορούμε να χρησιμοποιήσουμε και να εξάγουμε διαφορετικά συμπεράσματα. Τέλος θα δούμε το μοντέλο MediaPipe Pose, το οποίο αποτελεί μοντέλο μηχανική μάθησης και χρησιμοποιείται για την παρακολούθηση των χαρακτηριστικών του σώματος-πόζας ενός ατόμου.

3.2 Τεχνικές Ανίχνευσης Προσώπων

Όπως ήδη έχουμε αναφέρει η ανίχνευση προσώπου (face detection) έχει απασχολήσει τον τομέα την μηχανικής όρασης, ώστε να βρεθούν κατάλληλες τεχνικές για την επίτευξη αυτού του σκοπού. Η ανίχνευση προσώπου συνήθως αποτελείται από δύο φάσεις. Στην πρώτη φάση η εικόνα εξετάζεται έτσι ώστε να βρεθούν οι περιοχές

που ανιχνεύονται ως πρόσωπα (face detection), και στην δεύτερη φάση αφού εκτιμηθεί το μέγεθος ενός προσώπου, εκτελείται η διαδικασία για τον εντοπισμό προσώπου (face localization) όπου υπάρχει μια καλύτερη προσέγγιση για το που πραγματικά βρίσκεται το πρόσωπο στην εικόνα και το μέγεθος του.

Οι μέθοδοι και οι τεχνικές για την ανίχνευση προσώπου χωρίζονται τελικά σε δύο κύριες κατηγορίες (Εικόνα 3.1), βάση των προσεγγίσεων τους για τα χαρακτηριστικά του προσώπου. Η μια κατηγορία αφορά την ανίχνευση του προσώπου βάση των χαρακτηριστικών (Feature Based Approach) και η άλλη την ανίχνευση του προσώπου βάση την εμφάνιση (Image Based Approach).



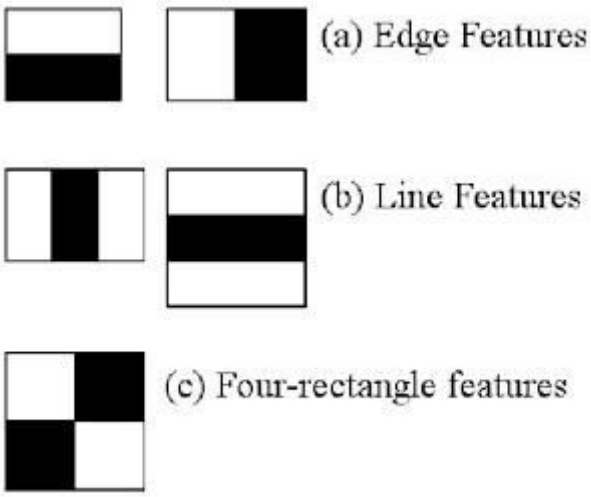
Εικόνα 3.1 Κατηγορίες και υποκατηγορίες τεχνικών για τον εντοπισμό προσώπου **[10]**

3.2.1 Haar cascade

Οι μέθοδοι ανίχνευσης που χρησιμοποιούνε χαρακτηριστικά τύπου Haar και χρησιμοποιούνται σε ψηφιακές εικόνες για την αναγνώρισης αντικειμένων, αποτελούν από τους πρώτους ανιχνευτές προσώπου που χρησιμοποιήθηκαν για την λειτουργία τους σε πραγματικό χρόνο. Η ιδέα προτάθηκε από τους Paul Viola και Michael Jones (2001) στην επιστημονική εργασία με τίτλο «Rapid Object Detection using a Boosted Cascade of Simple Features». Αποτελεί μια προσέγγιση όπου βασίζεται στην μηχανική μάθηση όπου, η λειτουργία cascade εκπαιδεύεται από πολλά θετικά και αρνητικά δείγματα.

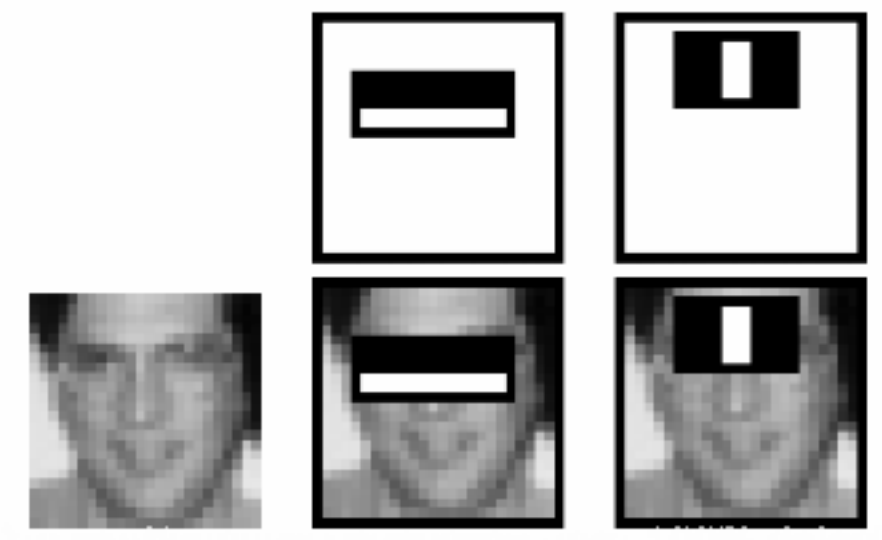
Στην περίπτωση μας θα δούμε πως λειτουργεί η μέθοδος για την ανίχνευση προσώπων. Ένα χαρακτηριστικό τύπου Haar λαμβάνει υπόψη τις γειτονικές ορθογώνιες

περιοχές σε μια συγκεκριμένη θέση της εικόνας σε ένα παράθυρο ανίχνευσης, λαμβάνει τις τιμές των pixel της κάθε περιοχής και έπειτα υπολογίζει την διαφορά μεταξύ των αθροισμάτων αυτών. Κάθε χαρακτηριστικό δηλαδή είναι μια ενιαία τιμή που προκύπτει, αφαιρώντας το άθροισμα των pixels από το μαύρο ορθογώνιο, από το άθροισμα των τιμών που υπάρχουν στο λευκό ορθογώνιο (Εικόνα 3.2). Η διαφορά αυτή που έχει υπολογιστεί, χρησιμοποιείται για την κατηγοριοποίηση των υποτμημάτων μια εικόνας. [11]



Εικόνα 3.2 5 διαφορετικά είδη χαρακτηριστικών τύπου Haar **[11]**

Σχετικά με την αναγνώριση προσώπου, το πιο κοινό χαρακτηριστικό σε ένα πρόσωπο, είναι η περιοχή των ματιών όπου τείνει να είναι πιο σκούρα από την περιοχή των μάγουλων και την περιοχή της μύτης, και το άλλο χαρακτηριστικό βασίζεται στην ιδιότητα ότι τα μάτια είναι πιο σκούρα από το πάνω μέρος της μύτης (την <<γέφυρα>> της μύτης). Στην Εικόνα 3.3 βλέπουμε αυτά τα Haar χαρακτηριστικά να επικεντρώνονται στις δύο περιοχές του προσώπου που αναφέραμε.



Εικόνα 3.3 2 χαρακτηριστικά τύπου Haar στο πρόσωπο **[11]**

Όλα τα πιθανά μεγέθη και οι τοποθεσίες κάθε πυρήνα, χρησιμοποιούνται για τον υπολογισμό πολλών χαρακτηριστικών (Ακόμα και ένα παράθυρο 24x24 έχει πάνω από 160000 χαρακτηριστικά). Προφανώς, αυτό αποτελεί ένα τεράστιο αριθμό, και για να λυθεί αυτό το πρόβλημα, εισήγαγαν τις ολοκληρωμένες εικόνες (intergral images) ή αλλιώς πίνακες αθροιστικής περιοχής, όπου προσφέρουν καλύτερα αποτελέσματα σε συνδυασμό με την χρήση του αλγορίθμου AdaBoost.

Τα πλεονεκτήματα των μεθόδων με χαρακτηριστικά τύπου Haar είναι ότι:

- Είναι πολύ γρήγορη, και δίνουν την δυνατότητα να τρέχουν σε συστήματα που απαιτούν την εφαρμογή τους σε πραγματικό χρόνο.
- Μπορούν να τρέξουν σε συστήματα με χαμηλές προδιαγραφές υλικού (όπως το Raspberry Pi).
- Το μέγεθος του μοντέλου είναι αρκετά μικρό (περίπου στα ~400KB).

Τα μειονεκτήματα των μεθόδων με χαρακτηριστικά τύπου Haar είναι ότι:

- Έχουν μεγάλο αριθμό σφαλμάτων, δηλαδή σε μια αναγνώριση προσώπου πολλές φορές εντοπίζουν αντικείμενα που δεν είναι πρόσωπα.
- Η στάση του κεφαλιού και του σώματος παίζουν ρόλο, καθώς υπάρχει μεγάλη πιθανότητα να μην πραγματοποιείται ο εντοπισμός.
- Δεν αποτελείται αξιόπιστη μέθοδος.

Καταληκτικά, η χρήση μεθόδων ανίχνευσης με χαρακτηριστικά τύπου Haar, δεν αποτελούν σε μεγάλο βαθμό αξιόπιστη μέθοδο για τον εντοπισμό προσώπου όταν η

χρήση που θέλουμε να πραγματοποιήσουμε απαιτεί την συνεχής αξιόπιστη παρακολούθηση προσώπου, αλλά δίνουν κατά κύριο λόγο καλύτερη απόδοση στο σύστημα, καθώς τρέχουν αρκετά γρήγορα. [12]

3.2.2 Caffe model

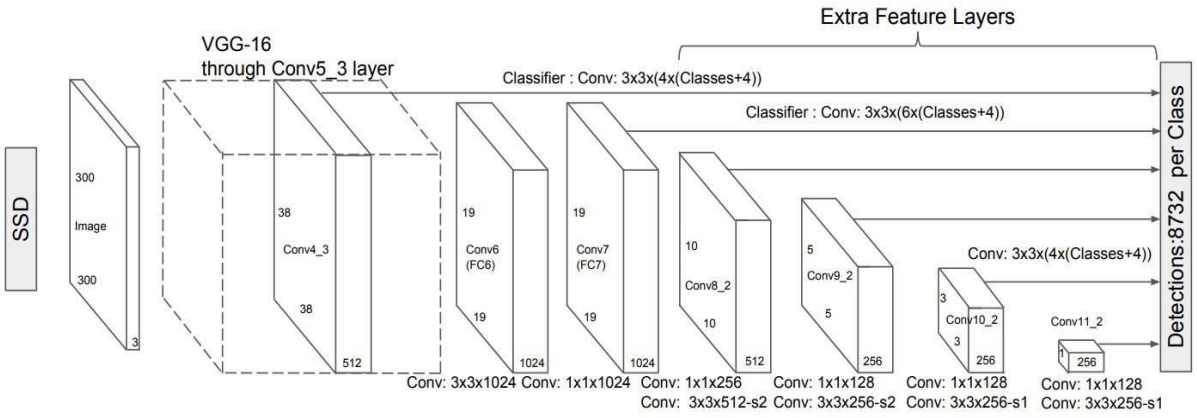
Το Caffe (Convolutional Architecture for Fast Feature Embedding) model, είναι ένα πλαίσιο (framework) βαθιάς μηχανικής μάθησης, όπου έχει ως βάση τον ανιχνευτή SSD (Single Shot Multibox Detector) και χρησιμοποιεί την αρχιτεκτονική ResNet-10. Το framework δημιουργήθηκε από τον Yangqing Jia κατά την διάρκεια της διδακτορικής του διατριβής στο πανεπιστήμιο του Berkeley.

Υποστηρίζει διαφορετικούς τύπους αρχιτεκτονικών βαθιά μηχανικής μάθησης που βρίσκονται προς την κατεύθυνση της ταξινόμησης εικόνων και την τμηματοποίηση εικόνων. Υποστηρίζει διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων όπως CNN, RCNN, LSTM και πλήρως συνδεδεμένα νευρωνικά δίκτυα. Επίσης δίνει την δυνατότητα υποστήριξης βιβλιοθηκών πυρήνα επιτάχυνσης που βασίζονται σε GPU και CPU (όπως NVDIA, cuDNN & Intel MKL). [13]

Όταν θέλουμε να χρησιμοποιήσουμε το βαθύ νευρωνικό δίκτυο (DNN-Deep Neural Network) του OpenCV με μοντέλα Caffe, θα πρέπει να συμπεριλάβουμε δύο τύπους αρχείων:

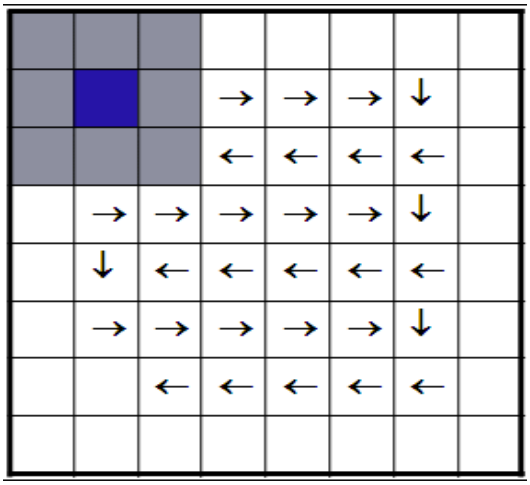
- Το **.prototxt** αρχείο το οποίο ορίζει την αρχιτεκτονική του μοντέλου.
- Το **.caffeemodel** αρχείο το οποίο αφορά τα βάρη που θα χρησιμοποιήσουν τα επίπεδα (layers) του μοντέλου.

Για να εμβαθύνουμε λίγο και να δούμε πως λειτουργεί το framework του Caffe Model, ας μιλήσουμε αρχικά για τον ανιχνευτή SSD. Ο ανιχνευτής SSD (Single Shot Multibox Detector), αποτελεί ένα ενοποιημένο πλαίσιο (unified framework) όπου χρησιμοποιείται για την ανίχνευση αντικειμένων χρησιμοποιώντας ένα μοναδικό δίκτυο. Αποτελείται από δύο στοιχεία, το πρώτο στοιχείο που το προσδιορίζει είναι το μοντέλου κορμού (Backbone Model) και το δεύτερο στοιχείο η κεφαλή SSD (SSD head). Το μοντέλο κορμού είναι συνήθως κάποιο προ-εκπαιδευμένο δίκτυο ταξινόμησης εικόνων (image classification network), το οποίο δίκτυο συνήθως μπορεί να είναι το ResNet εκπαιδευμένο στο δίκτυο ImageNet. Στην Εικόνα 3.4 βλέπουμε την αρχιτεκτονική του SSD, βασιζόμενη στο μοντέλο του συνελικτικού νευρωνικού δικτύου VGG.



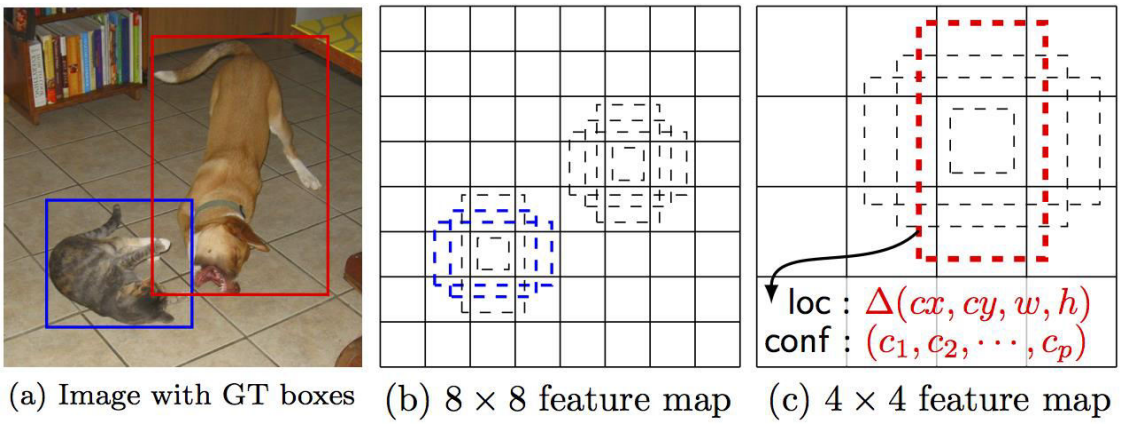
Εικόνα 3.4 Αρχιτεκτονική SSD **[14]**

Ο ανιχνευτής SSD δεν χρησιμοποιεί το κλασικό αλγόριθμό συρόμενου παραθύρου , όπου εκεί όπως φαίνεται στην Εικόνα 3.5, ένα πλαίσιο 3x3 χρειάζεται να διαπεράσει ολόκληρη την εικόνα με την σειρά που αναδεικνύεται (με την κατεύθυνση που έχουν τα βελάκια).



Εικόνα 3.5 Λειτουργία αλγορίθμου συρόμενου παραθύρου **[15]**

Σε αντίθεση, ο ανιχνευτής SSD διαιρεί την εικόνα ως πλέγματα και κάθε κελί πλέγματος είναι υπεύθυνο για την ανίχνευση αντικειμένων στην εκάστοτε περιοχή της εικόνας. Εάν στην περίπτωση υπάρχουν πολλά αντικείμενα στην εικόνα τότε χρησιμοποιούνται τα Anchor Boxes τα οποία είναι κουτιά με προκαθορισμένο μέγεθος και σχήμα μέσα στο κελί του πλέγματος (Εικόνα 3.6 (b), (c)).



Εικόνα 3.6 Τεχνική ανίχνευσης με χρήση SSD **[16]**

Αφού είδαμε συνοπτικά, τα βασικά χαρακτηριστικά του ανιχνευτή SSD, ας επιστρέψουμε ώστε να δούμε πλεονεκτήματα και μειονεκτήματα του ανιχνευτή προσώπου βαθιάς μηχανικής μάθησης του OpenCV (που βασίζεται στον ανιχνευτή SSD).

Τα πλεονεκτήματα του ανιχνευτή είναι:

- Μεγάλη ακρίβεια στην ανίχνευση προσώπου.
- Μη επιρρεπής σε μεγάλο ποσοστό, στην ανίχνευση αντικειμένου που δεν αφορά πρόσωπο.
- Δεν απαιτεί συντονισμό παραμέτρων.
- Μπορεί να τρέξει σε πραγματικό χρόνο, πάνω σε συστήματα με χαμηλές προδιαγραφές υλικού (όπως το Raspberry Pi).
- Το μέγεθος του μοντέλου δεν είναι πολύ μεγάλο (περίπου στα ~10MB).
- Μπορεί να εκτελεστεί ακόμη πιο γρήγορα, σε ενσωματωμένα συστήματα, χρησιμοποιώντας επιπλέον την διεπαφή OpenVIVO και το Movidius NCS (Movidius Neural Compute Stick) τα οποία σε συνδυασμό, επιτρέπουν τη γρήγορη δημιουργία προτύπων σε βαθιά νευρωνικά δίκτυα.

Τα μειονεκτήματα του ανιχνευτή είναι, ότι:

- Έχει μεγαλύτερη πολυπλοκότητα από ανιχνευτές που χρησιμοποιούνε χαρακτηριστικά τύπου Haar (Haar Cascades) [12]

3.3 Εντοπισμός Χαρακτηριστικών Προσώπου με την βιβλιοθήκη dlib

Η βιβλιοθήκη dlib, περιέχει ένα μεγάλο εύρος αλγορίθμων και εργαλείων μηχανικής μάθησης, οι οποίοι εφαρμόζονται για την δημιουργία λογισμικού και την επίλυση προβλημάτων σε τομείς όπως ρομποτική, όραση υπολογιστών, δίκτυα υπολογιστών, ενσωματωμένα συστήματα κτλ. Η βιβλιοθήκη dlib είναι γραμμένη σε C++, αλλά διαθέτει διεπαφή η οποία δίνει τη δυνατότητα πρόσβασης και στη γλώσσα Python. [17]

Η ανίχνευση των χαρακτηριστικών σημείων ενός προσώπου, αποτελεί ένα υποσύνολο του προβλήματος για την πρόβλεψη του σχήματος. Με βάση μια εικόνα εισόδου, το μοντέλο επιχειρεί να εντοπίσει τα βασικά χαρακτηριστικά σημεία του προσώπου. Συνήθως η διαδικασία είναι ότι πρώτα εντοπίζουμε την περιοχή του προσώπου με κάποιο μοντέλο, όπως είδαμε στις υποενότητες 3.2.1 και 3.2.2 (υπάρχουν και επιπλέον μέθοδοι για την ανίχνευση του προσώπου όπως MTCNN, HOG κτλ.). Έπειτα αφού έχουμε εντοπίσει την περιοχή του προσώπου, τη χρησιμοποιούμε ως είσοδο στο μοντέλο ανίχνευσης των χαρακτηριστικών του προσώπου (dlib 68 landmarks), ώστε να εντοπίσουμε τα χαρακτηριστικά του προσώπου.

Για την περίπτωση μας η βιβλιοθήκη dlib, διαθέτει ένα έτοιμο μοντέλο (shape_predictor_68_face_landmarks.dat), το οποίο δύναται να εντοπίσει 68 χαρακτηριστικά του προσώπου (Εικόνα 3.7).

Εικόνα 3.7 68 χαρακτηριστικά σημεία του προσώπου αριθμημένα, μέσω του μοντέλου της dlib βιβλιοθήκης **[18]**

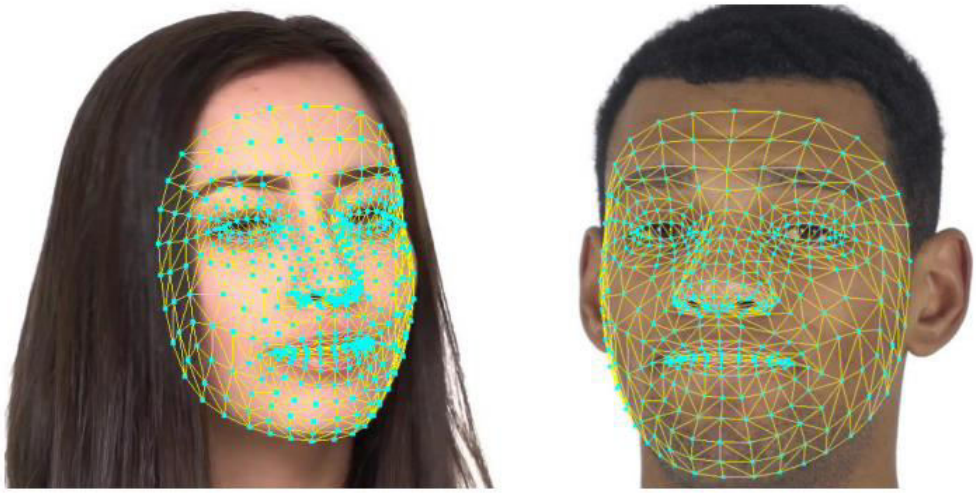
Τα χαρακτηριστικά της Εικόνας 3.7 χωρίζονται σε 8 συγκεκριμένες κατηγορίες, και ομαδοποιούνται βάση των αριθμών τους όπως βλέπουμε στον Πίνακα 1.

Χαρακτηριστικά Σημεία Προσώπου	**Αρίθμηση των σημείων του προσώπου**
Σαγόνι	0 έως 16
Δεξί φρύδι	17 έως 21
Αριστερό φρύδι	22 έως 26
Μύτη	27 έως 35
Δεξί μάτι	36 έως 41
Αριστερό μάτι	42 έως 47
Στόμα	48 έως 59
Εσωτερικό μέρος στόματος	60 έως 67

Πίνακας 1 Τα 68 χαρακτηριστικά σημεία του προσώπου ομαδοποιημένα ως προς την αρίθμηση τους

Εάν σε περίπτωση, θέλουμε να χρησιμοποιήσουμε κάποιο μοντέλο, που μας δίνει πολλά περισσότερα χαρακτηριστικά σημεία του προσώπου ώστε να τα χρησιμοποιήσουμε για την υλοποίηση κάποιας εφαρμογής που τα απαιτεί, τότε μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη MediaPipe όπου μας δίνει 468 χαρακτηριστικά σημεία του προσώπου. Η βιβλιοθήκη MediaPipe Face Mesh προσφέρει

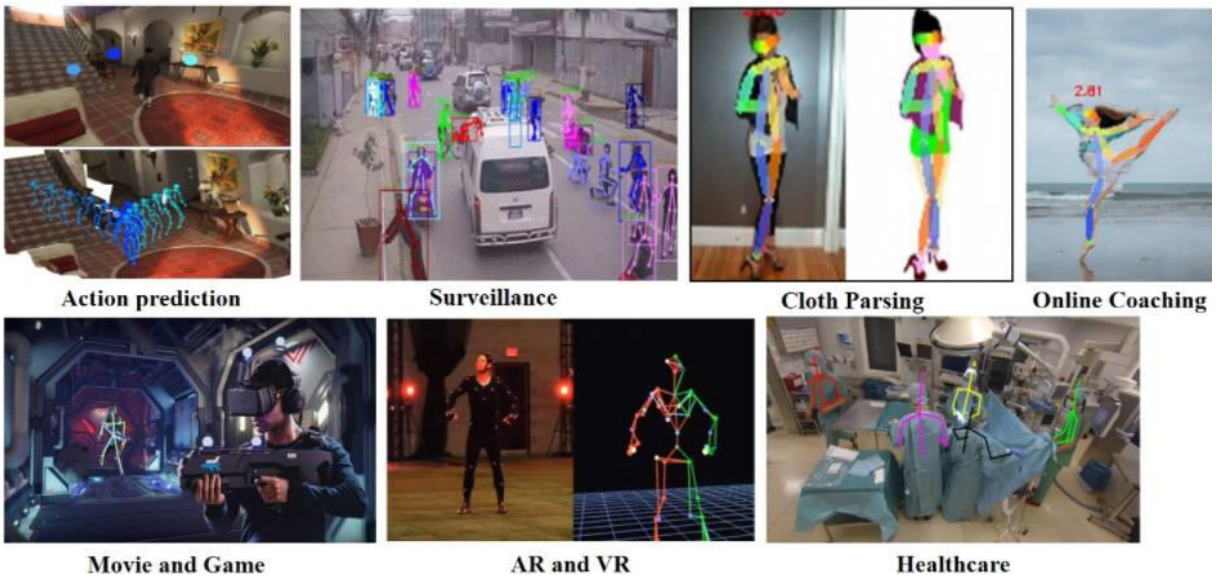
των εντοπισμό 468 χαρακτηριστικών σημείων του προσώπου (Εικόνα 3.8), και μπορεί επίσης να λειτουργήσει σε συστήματα που απαιτούν τον εντοπισμό αυτών των σημείων σε πραγματικό χρόνο, καθώς χρησιμοποιεί αρχιτεκτονικές και μοντέλα οι οποίες χρησιμοποιούν τη λιγότερο δυνατή πολυπλοκότητα για την εκτέλεση τους.



Εικόνα 3.8 Γραφική αναπαράσταση 468 σημείων σε τοπολογία πλέγματος **[19]**

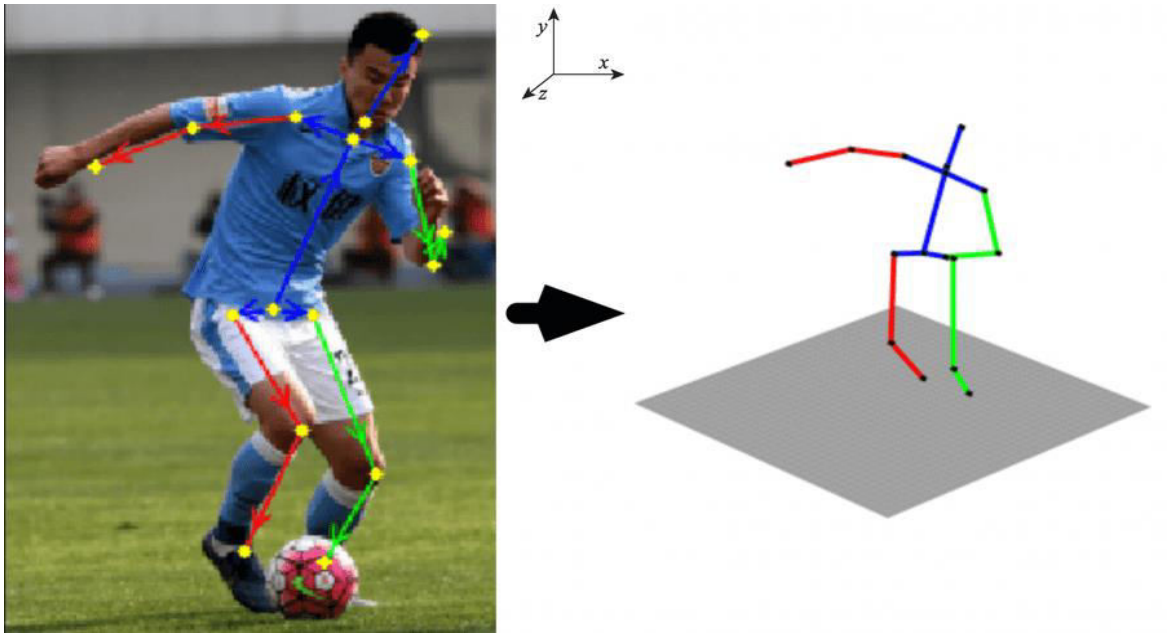
3.4 Εντοπισμός Χαρακτηριστικών Σώματος-Πόζας

Η τεχνική εντοπισμού των σημείων του σώματος-πόζας ενός ανθρώπου, αποτελεί μια τεχνική η οποία προβλέπει και παρακολουθεί τη θέση ενός ατόμου. Αυτό επιτυγχάνεται συνδυάζοντας την στάση και τον προσανατολισμό του σώματος ενός ατόμου. Με την τεχνική αυτή μας δίνεται η δυνατότητα να εξετάσουμε την κίνηση ενός (η πολλών) διαφορετικών ατόμων, δίνοντας την δυνατότητα, να μπορέσουμε να δημιουργήσουμε διάφορες εφαρμογές (όπως π.χ. εικονικούς προπονητές για διάφορα αθλήματα και δραστηριότητες, διασφάλιση ασφάλειας με βάση τις κινήσεις των ατόμων, αναγνώριση της <<γλώσσας του σώματος>> με βάση τις κινήσεις του ατόμου κτλ. όπως βλέπουμε και στην Εικόνα 3.9).



Εικόνα 3.9 Εφαρμογές του εντοπισμού της πόζας **[20]**

Υπάρχει μια βασική διαφορά μεταξύ του εντοπισμού των σημείων του σώματος σε δισδιάστατη μορφή (2D) και σε τρισδιάστατη μορφή (3D), όπως μπορούμε να δούμε και στην Εικόνα 3.10.

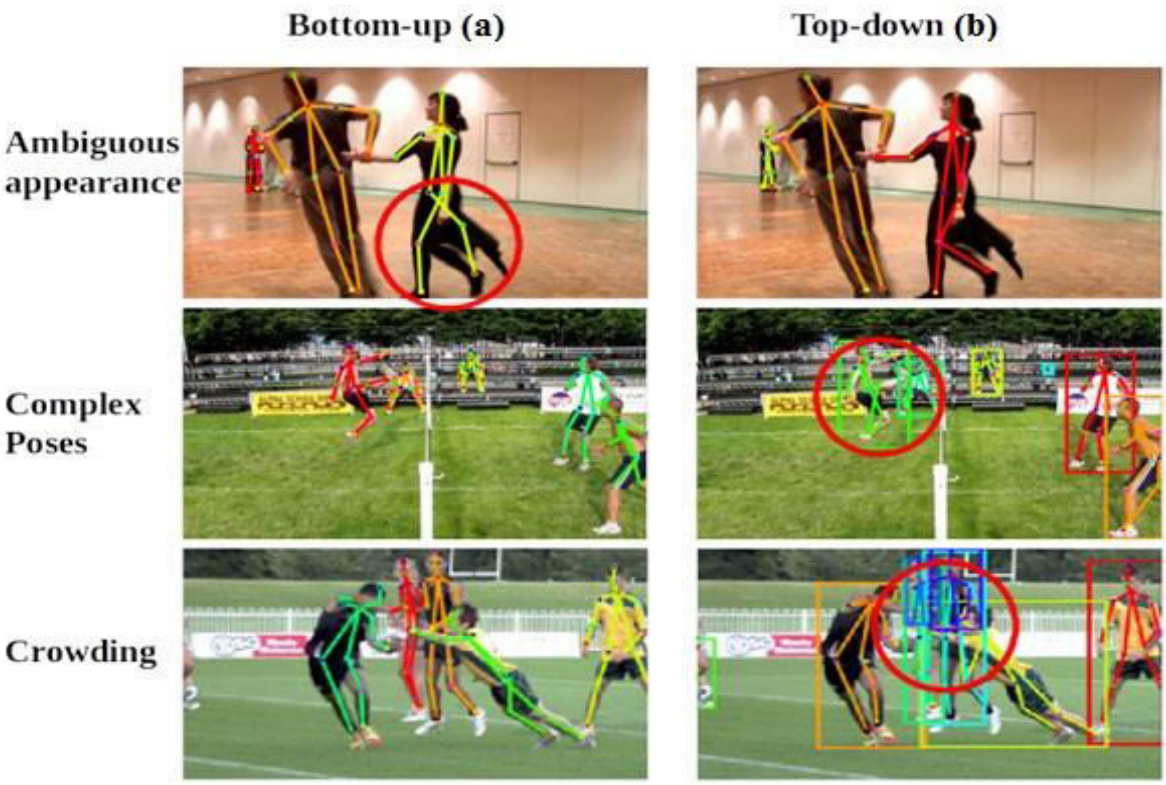


Εικόνα 3.10 2D αναπαράσταση στα αριστερά και η αντίστοιχη 3D αναπαράσταση στα δεξιά **[21]**

Ο εντοπισμός σε δισδιάστατη μορφή, βρίσκει τη θέση των σημείων στο δισδιάστατο χώρο σε σχέση με μια εικόνα, και υπολογίζει τις συντεταγμένες X και Y για κάθε σημείο του σώματος του ατόμου. Στην περίπτωση του τρισδιάστατου εντοπισμού, χρησιμοποιείται το αντικείμενο της δισδιάστατης εικόνας, ώστε να μετατραπεί σε αντικείμενο τρισδιάστατης εικόνας προσθέτοντας επιπλέον μια διάσταση Z. Ο

τρισδιάστατος εντοπισμός, επιτρέπει να προβλέψουμε την πραγματική χωρική τοποθέτηση ενός ατόμου, αλλά για να επιτευχθεί απαιτείται μεγαλύτερη υπολογιστική πολυπλοκότητα.

Οι αρχιτεκτονικές βαθιάς μηχανικής μάθησης, είναι κατάλληλες για την εντοπισμό των χαρακτηριστικών του σώματος, και βασίζονται συνήθως σε παραλλαγές συνελικτικών νευρωνικών δικτύων (CNN). Γενικά υπάρχουν δύο τρόποι προσέγγισης για τον εντοπισμό των χαρακτηριστικών του σώματος, σε εικόνες όπου υπάρχουν τουλάχιστον δύο άτομα και άνω. Η πρώτη προσέγγιση, είναι η από κάτω προς τα πάνω (bottom-up approach), η οποία αρχικά ανιχνεύει τα χαρακτηριστικά σημεία του σώματος για κάθε ομάδα ξεχωριστά (π.χ. όλα τα δεξιά χέρια) και ύστερα τα ομαδοποιεί, έτσι ώστε να σχηματιστεί μια μοναδική αναπαράσταση του σώματος. Η δεύτερη προσέγγιση είναι η από πάνω προς τα κάτω (top-down approach), η οποία αρχικά ανιχνεύει την περιοχή του κάθε ατόμου που βρίσκεται στην εικόνα, με έναν ανιχνευτή ατόμου (person detector), και ύστερα για την κάθε περιοχή εντοπίζει τα χαρακτηριστικά σημεία του σώματος κάθε ατόμου ξεχωριστά. Η προσέγγιση από κάτω προς τα πάνω (bottom-up approach), είναι καλύτερη σε σύνθετες στάσεις του σώματος, αλλά πολλές φορές παρουσιάζει ψευδώς θετικά αποτελέσματα (Εικόνα 3.11 (a)). Από την άλλη η προσέγγιση από πάνω προς τα κάτω (top-down approach), βασίζεται αρχικά στον εντοπισμό του ατόμου στην εικόνα, και έπειτα στην εύρεση των χαρακτηριστικών του. Επομένως, η προσέγγιση αυτή, είναι πιθανόν σε πολύπλοκες στάσεις του σώματος του ατόμου, να μην πραγματοποιεί καθόλου εντοπισμούς(Εικόνα 3.11 (b)).



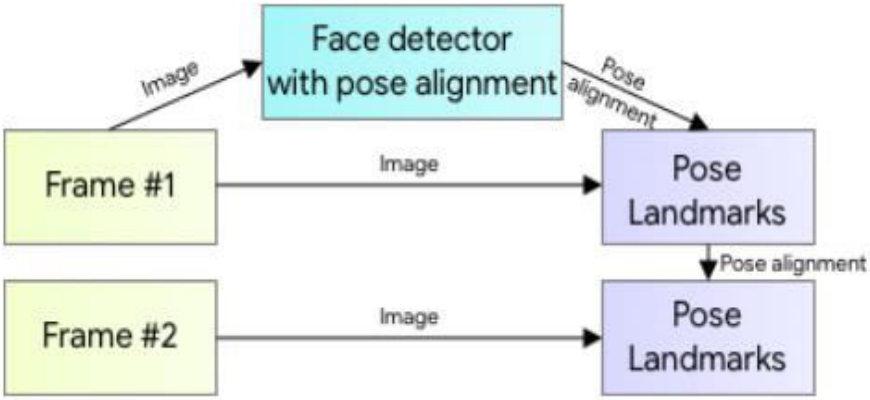
Εικόνα 3.11 (a)Τεχνική από κάτω προς τα πάνω & (b)Τεχνική από πάνω προς τα κάτω **[22]**

3.4.1 Εντοπισμός Χαρακτηριστικών σημείων σώματος με χρήση της βιβλιοθήκης MediaPipe

Η MediaPipe αποτελεί ένα πλαίσιο λογισμικού (framework), το όποιο δίνει την δυνατότητα ανάπτυξης πολύπλοκών εφαρμογών, οι οποίες έχουν ανάγκη να χρησιμοποιήσουν πολύπλοκούς αλγορίθμους μηχανικής μάθησης και μηχανικής όρασης. Περιέχει λύσεις σε προβλήματα, που αφορούν την ανίχνευση προσώπου, τον εντοπισμό πλέγματος του προσώπου (υπολογίζοντας 468 σημεία το προσώπου), την ανίχνευση αντικειμένων, την άμεση παρακολούθηση κίνησης κτλ. Μια επιπλέον λύση που προσφέρει και θα ασχοληθούμε στο έργο μας είναι ο εντοπισμός των χαρακτηριστικών σημείων του σώματος.

Η MediaPipe Pose αφορά τον εντοπισμό 33 χαρακτηριστικών σημείων του σώματος σε τρισδιάστατή μορφή, και βασίζεται στην επιστημονική εργασία BlazePose που πραγματοποίησαν ερευνητές της Google. Η προσέγγιση που ακολουθεί αυτή η λύση χωρίζεται σε δύο βασικά βήματα. Το πρώτο βήμα, που αποτελεί τον εντοπισμού του ατόμου στο εκάστοτε πλαίσιο, και την εξαγωγή της περιοχής ενδιαφέροντος (ROI-Region Of Interest). Στην συνέχεια στο δεύτερο βήμα, με βάση την περιοχή ενδιαφέροντος

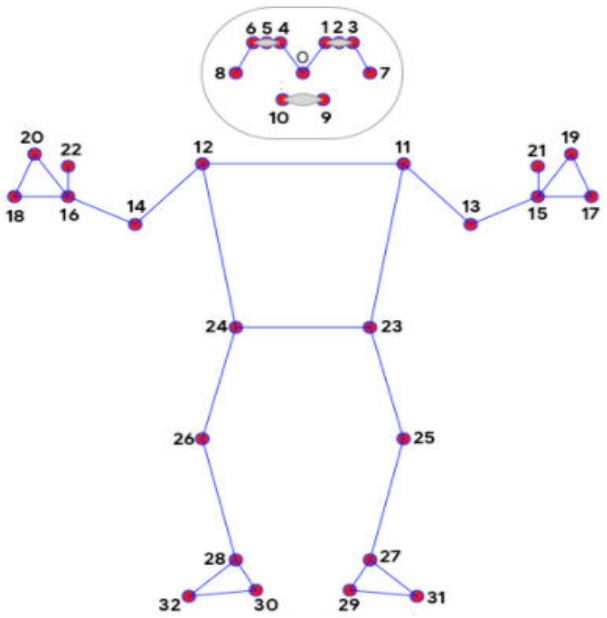
(όπου έχουμε εντοπίσει το άτομο) πραγματοποιείται ο εντοπισμός των χαρακτηριστικών σημείων του ατόμου. Στην Εικόνα 3.12 βλέπουμε την ροή της διαδικασίας που ακολουθείται.



Εικόνα 3.12 Διαδικασία εύρεσης χαρακτηριστικών σώματος **[23]**

Σε μια ροή βίντεο, ο ανιχνευτής για τον εντοπισμό του ατόμου, καλείτε μόνο στο πρώτο πλαίσιο (με την προϋπόθεση ότι το άτομο έχει εντοπισθεί στο πρώτο πλαίσιο), και ύστερα εντοπίζονται τα χαρακτηριστικά σημεία του σώματος του. Έπειτα από την στιγμή που το άτομο έχει ήδη εντοπισθεί στο πρώτο πλαίσιο του βίντεο, τότε στο επόμενο πλαίσιο δεν χρειάζεται να ξανά εντοπισθεί, απλά περνάει ως είσοδος και ακολουθείται η ίδια διαδικασία για τον εντοπισμό των χαρακτηριστικών.

Στην Εικόνα 3.13 βλέπουμε τα 33 χαρακτηριστικά σημεία που δύναται να εντοπίσει το μοντέλο.



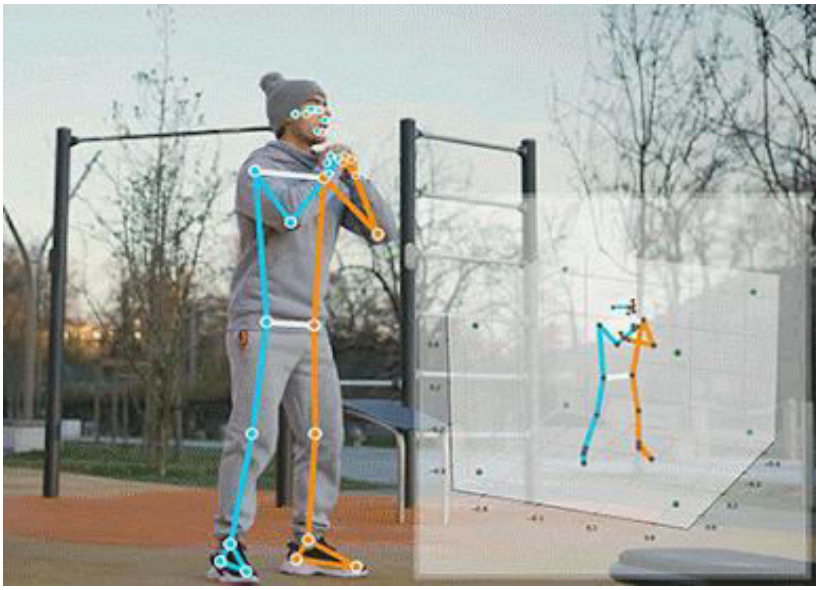
0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Εικόνα 3.13 33 χαρακτηριστικά σημεία του σώματος **[24]**

Στα αριστερά βλέπουμε τα σημεία του σώματος αριθμημένα, και στα δεξιά βλέπουμε την αντιστοίχηση των αριθμών με τα συγκεκριμένα χαρακτηριστικά του σώματος. Αν ομαδοποιήσουμε τα χαρακτηριστικά της Εικόνας 3.13, τότε έχουμε 10 γενικές κατηγορίες, οι οποίες ομαδοποιούνται όπως βλέπουμε στον Πίνακα 2.

Χαρακτηριστικά Σημεία	**Αρίθμηση των σημείων**
Μύτη	0
Αριστερό & Δεξί μάτι	1 έως 3 & 4 έως 6
Αριστερό & Δεξί αυτί	7 & 8
Στόμα	9 έως 10
Αριστερός & Δεξιός ώμος	11 & 12
Αριστερός & Δεξιός αγκώνας	13 & 14
Αριστερό & Δεξί χέρι	15,17,19,21 & 16,18,20,22
Αριστερό & Δεξιό σημείο στη μέση	23 & 24
Αριστερό & δεξί γόνατο	25 & 26
Αριστερό & δεξί πόδι	27,29,31 & 28,30,32

Πίνακας 2 Τα 33 χαρακτηριστικά σημεία του σώματος ομαδοποιημένα



Εικόνα 3.14 2D & 3D Αναπαράσταση σημείων του σώματος με χρήση της βιβλιοθήκης MediaPipe **[24]**

4. Περιγραφή Συστήματος Υλικού

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα περιγράψουμε και θα συναρμολογήσουμε όλο το απαραίτητο υλικό, το οποίο χρησιμοποιήθηκε για να υλοποιηθεί το έργο μας. Αρχικά, θα δούμε συνοπτικά όλα τα μέρη του υλικού μας. Έπειτα, θα αναλύσουμε τα χαρακτηριστικά του υλικού που χρησιμοποιήθηκαν. Τέλος θα δούμε με βήμα-βήμα, πως μπορούμε να συνδυάσουμε όλο το απαραίτητο υλικό, ώστε να στήσουμε σωστά το Raspberry Pi σε συνδυασμό με το εξωτερικό υλικό μας.

4.2 Συνοπτικά το Υλικό

Παρακάτω βλέπουμε συνοπτικά το υλικό που χρησιμοποιήθηκε για την υλοποίηση του έργου μας:

1. Raspberry Pi 4 Model B (2GB RAM)
2. Κάρτα Μνήμης microSDHC 32GB Class 10 SanDisk Ultra
3. 3 ψύκτρες για το Raspberry Pi 4 (3 Heat Sinks)
4. Ανεμιστηράκι (DC Fan 30x70x7mm 5V)
5. Θήκη (Oval Case)
6. Τροφοδοτικό (5 Volt, 3 Ampere, USB Type C)
7. Κάμερα Υπερύθρων (5MP, 1080p)
8. Πειραματική Διάταξη 830 οπές (Breadboard 830 Points)
9. GPIO Πίνακας Επέκτασης (GPIO Extension Board)
10. A9G GSM/GPRS+GPS Module
11. 3 Δίοδοι φωτοεκπομπής (LED)(1 Πράσινο, 1 Κίτρινο, 1 Κόκκινο)
12. 3 Αντιστάσεις των 220Ω η κάθε μια
13. 12 Καλώδια Βραχυκυκλωτήρα Αρσενικό σε Αρσενικό (12 Jumper Wires M-M)

4.3 Αναλυτικά το Υλικό

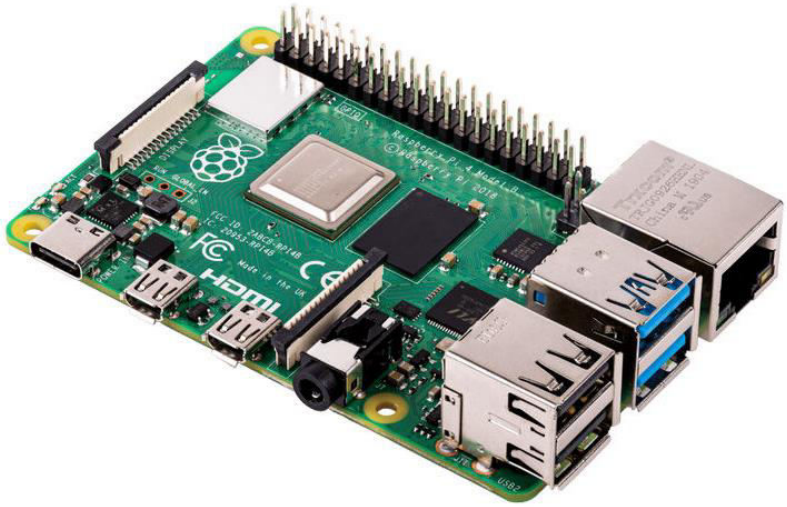
Παρακάτω βλέπουμε πιο αναλυτικά το υλικό που χρησιμοποιήθηκε για την υλοποίηση του έργου μας:

1. Raspberry Pi 4 Model B (2GB RAM):

Το Raspberry Pi είναι μια σειρά υπολογιστών χαμηλού κόστους, μικρού μεγέθους τα οποία δίνουν την δυνατότητα χρήσης τους, σαν ένα κοινό υπολογιστικό σύστημα. Τα Raspberry Pi, αναπτύχθηκαν στο Ηνωμένο Βασίλειο, από το Raspberry Pi Foundation σε συνεργασία με την Broadcom.

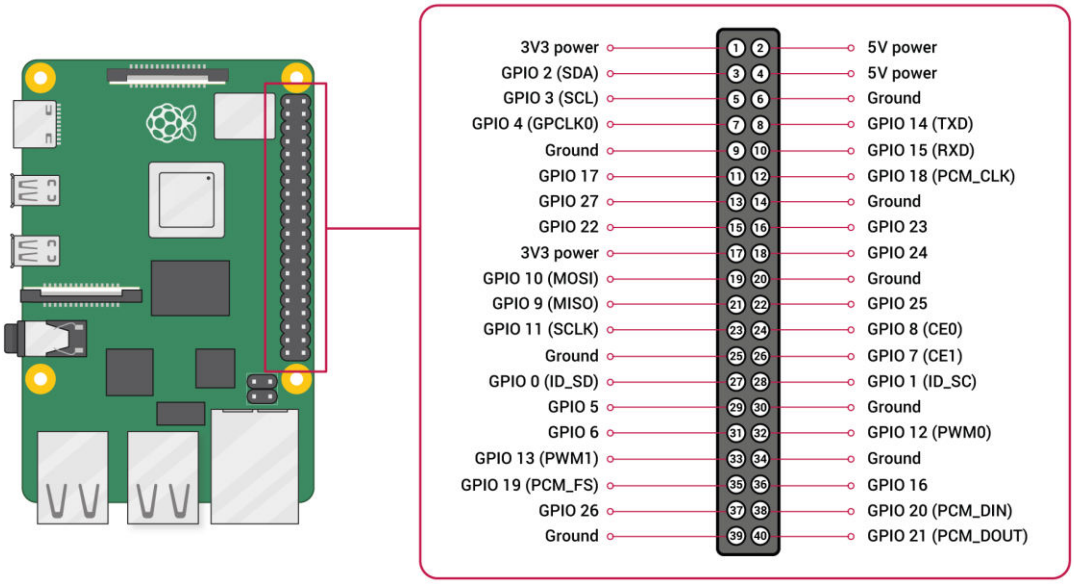
Άρχισαν να κυκλοφορούν το 2012, με πρώτο μοντέλο το Raspberry Pi 1 model B και από την αρχή φάνηκε να υπήρχε άμεση ανταπόκριση από τον κόσμο για την ευέλικτη χρήση τους. Σημειώνοντας μεγάλη επιτυχία, τα Raspberry Pi από το 2012 και έκτοτε έχουν πουληθεί σε τουλάχιστον 40 εκατομμύρια κομμάτια (2021).

Το Raspberry Pi 4 Model B κυκλοφόρησε τον Ιούνιο του 2019, και προσφέρει μεγαλύτερη επεξεργαστική ισχύ, περισσότερη μνήμη και γενικά έρχεται πιο βελτιωμένο σε σχέση με τα προηγούμενα μοντέλα (Εικόνα 4.1). [25]



Εικόνα 4.1 Raspberry Pi 4 Model B (2GB RAM)

Ένα ακόμη σημαντικό χαρακτηριστικό του Raspberry Pi είναι το GPIO, το οποίο βρίσκεται πάνω στην πλακέτα, και αποτελείται από 40 PIN (Εικόνα 4.2). Η δυνατότητα που μας δίνεται είναι πως μπορούμε να χρησιμοποιήσουμε συγκεκριμένα GPIO pin και να τα διαχειριστούμε μέσω λογισμικού.



Εικόνα 4.2 40 GPIO PIN Raspberry Pi **[26]**

Ας δούμε πως χωρίζονται τα 40 GPIO PIN του Raspberry Pi 4 Model B:

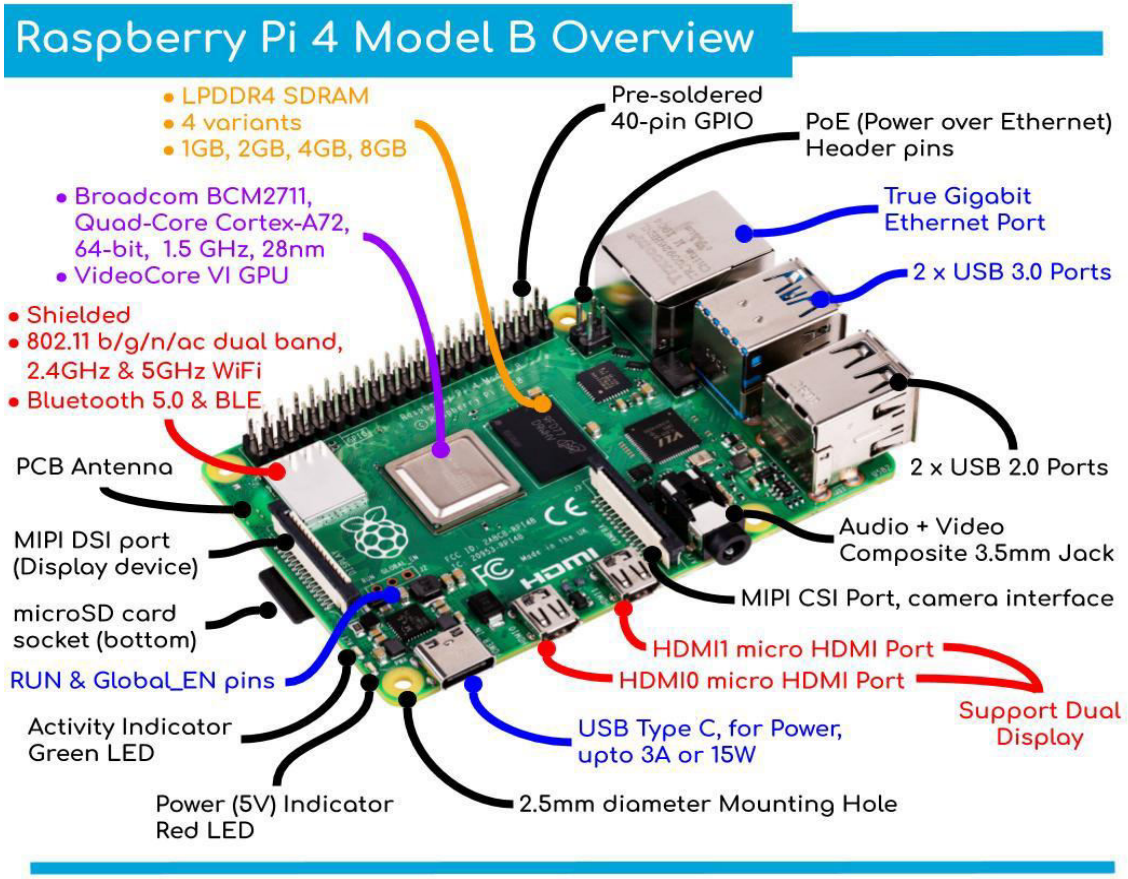
- 26 ακροδέκτες Γενικού Σκοπού (GPIO)
- 8 ακροδέκτες Γείωσης (Ground)
- 2 ακροδέκτες 5V
- 2 ακροδέκτες 3V3
- 2 ακροδέκτες ID EEPROM

Το Raspberry Pi 4 Model B αποτελείται από τα παρακάτω χαρακτηριστικά σε υλικό (Εικόνα 4.3):

- Τετραπύρηνος Επεξεργαστής 64-bit ARM-Cortex A72 με επεξεργαστική ισχύ 1.5GHz
- 2GB LPDDR4 RAM (Υπάρχουν Raspberry Pi με 4GB & 8GB LPDDR4 RAM)
- H.265 (HEVC) Αποκωδικοποίηση έως και 4Kp60
- H.264 Αποκωδικοποίηση έως και 1080p60
- VideoCore VI 3D Graphics
- Υποστηρίζει διπλή HDMI απεικόνιση έως και 4Kp60

Το Raspberry Pi 4 Model B περιέχει τις παρακάτω διεπαφές (Εικόνα 4.3):

- 2.4GHZ και 5.0GHz IEEE 802.11 b/g/n/ac Ασύρματο τοπικό δίκτυο (WLAN)
- LAN, Bluetooth 5.0 με BLE
- 1x microSD Card
- 2x mirco-HDMI ports οι οποίες υποστηρίζουν απεικόνιση έως και 4Kp60
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port
- 1x Raspberry Pi Camera port (MIPI CSI)
- 1x Raspberry Pi Display port (MIPI DSI)
- 1x 4-pole stereo audio
- 40-pin GPIO header



Εικόνα 4.3 Χαρακτηριστικά και διεπαφές του Raspberry Pi 4 Model B **[27]**

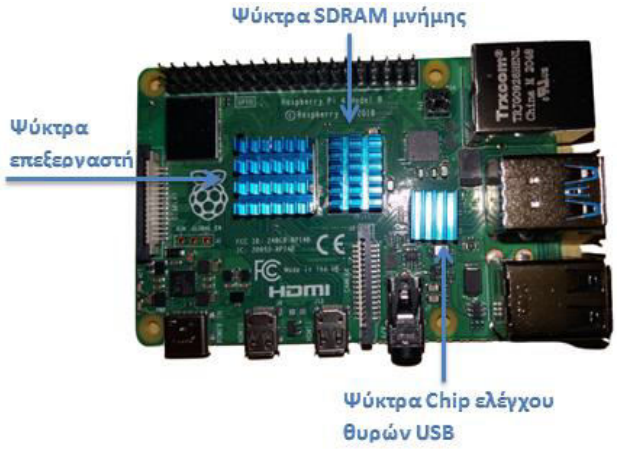
2. **Κάρτα Μνήμης microSDHC 32GB Class 10 SanDisk Ultra:**

Κάρτα Μνήμης με χωρητικότητα 32GB ή οποία χρησιμοποιήθηκε για την εγκατάσταση του λειτουργικού συστήματος Raspberry Pi OS και την χρήση κατά την διάρκεια του έργου (Εικόνα 4.4).

Εικόνα 4.4 Κάρτα Μνήμης microSDHC 32GB Class 10 SanDisk Ultra

3. **3 ψύκτρες για το Raspberry Pi 4 (3 Heat Sinks)**

Οι ψύκτρες χρησιμοποιήθηκαν για την ψύξη των ηλεκτρονικών εξαρτημάτων του Raspberry Pi και συγκεκριμένα για την ψύξη του επεξεργαστή, της SDRAM και του chip που αφορά τον έλεγχο των USB θυρών (Εικόνα 4.5).



Εικόνα 4.5 Ψύκτρες στο Raspberry Pi

4. **Ανεμιστηράκι (DC Fan 30x70x7mm 5V)**

Το ανεμιστηράκι χρησιμοποιήθηκε για την καλή ροή αέρα μέσα στην θήκη και την ψύξη των εξαρτημάτων μέσα το Raspberry Pi (Εικόνα 4.6).

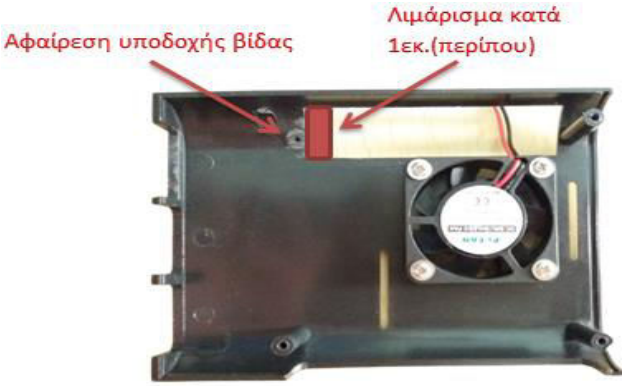
Εικόνα 4.6 Ανεμιστηράκι Raspberry Pi

5. **Θήκη (Oval Case)**

Χρησιμοποιήθηκε έτσι ώστε να διασφαλίζει την καλύτερη ροή αέρα, μαζί με το ανεμιστηράκι και να αποτρέπει το Raspberry Pi, από την έκθεση του στην σκόνη (Εικόνα 4.7).

Εικόνα 4.7 Θήκη Raspberry Pi

Προσοχή! Εάν θέλουμε να χρησιμοποιήσουμε την συγκεκριμένη θήκη μαζί με GPIO Πίνακας Επέκτασης 40 ακροδεκτών (GPIO Extension Board 40 pin), θα πρέπει να κόψουμε την υποδοχή για την βίδα δίπλα από τους ακροδέκτες, και να λιμάρουμε λίγο, το αριστερό μέρος της θήκης όπως βλέπουμε στην Εικόνα 4.8.



Εικόνα 4.8 Τροποποιημένη θήκη Raspberry Pi

6. **Τροφοδοτικό (5 Volt, 3 Ampere, USB Type C)**

Τροφοδοτικό κατάλληλο για την τροφοδοσία του Raspberry Pi (Εικόνα 4.9).

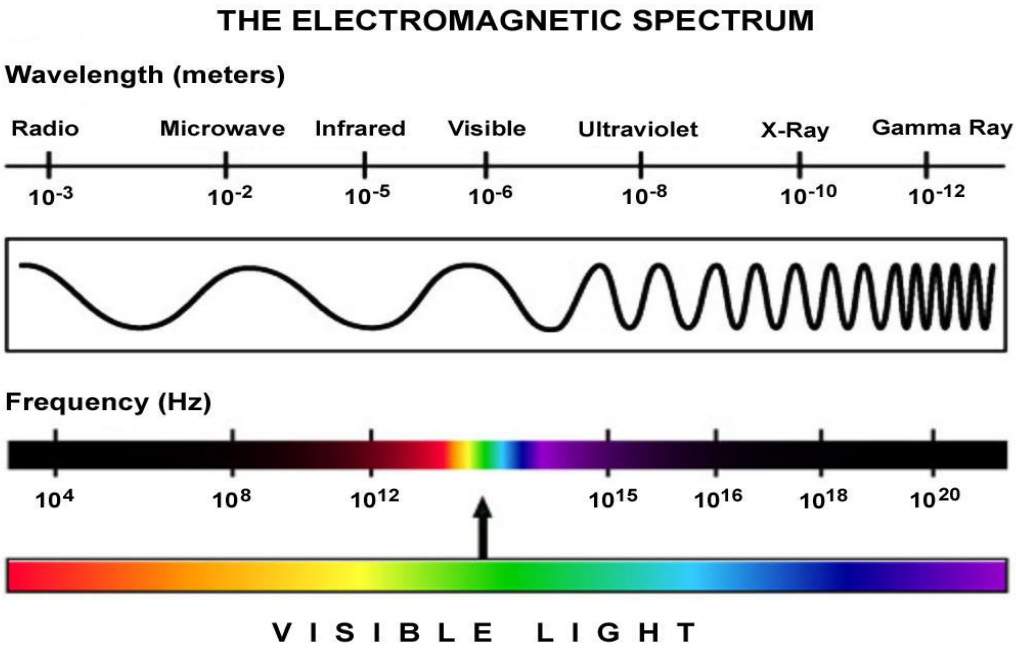
Εικόνα 4.9 Τροφοδοτικό Raspberry Pi

7. Κάμερα Νυχτερινής Λήψης (Night Vision Camera 5MP, 1080p)

Η κάμερα νυχτερινής λήψης (night vision camera) ή αλλιώς κάμερα υπερύθρων (IR camera), ήταν απαραίτητη για την δημιουργία του έργου μας, καθώς ο κύριος στόχος του συστήματος μας, ήταν να μπορούμε να παρακολουθούμε τις κινήσεις και τα χαρακτηριστικά του οδηγού και στο σκοτάδι, δηλαδή και σε καταστάσεις τις οποίες δεν υπάρχει ορατό φως (Εικόνα 4.10).

Εικόνα 4.10 Κάμερα Υπερύθρων για το Raspberry Pi

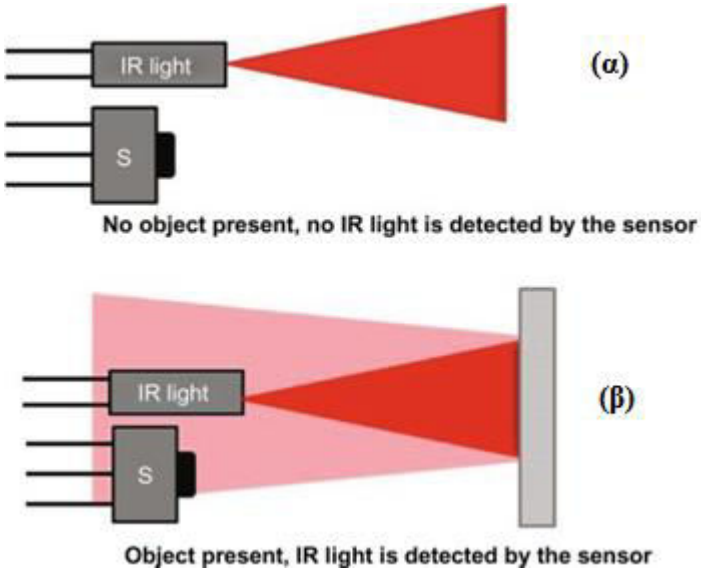
Για να δούμε πως λειτουργεί μια κάμερα υπερύθρων, θα πρέπει αρχικά να δούμε το ηλεκτρομαγνητικό φάσμα το οποίο χωρίζεται σε επιμέρους ζώνες. Αυτές οι ζώνες είναι τα ραδιοκύματα (10^{-3}), τα μικροκύματα (10^{-2}), η υπέρυθρη ακτινοβολία(10^{-5}), το ορατό φως(10^{-6}), η υπεριώδης ακτινοβολία(10^{-8}), οι ακτίνες X (10^{-10}) και οι ακτίνες γάμμα (10^{-12}) (Εικόνα 4.11).



Εικόνα 4.11 7 ζώνες του ηλεκτρομαγνητικού φάσματος **[28]**

Η κατηγορία του υπέρυθρου φωτός, φαίνεται να βρίσκεται σε χαμηλότερη συχνότητα από τα ορατά χρώματα που μπορεί να διακρίνει ένας άνθρωπός. Οι

κάμερες νυχτερινής λήψεως, χρησιμοποιούν εκπομπούς (IR Transmitter) η οποίοι εκπέμπουν υπέρυθρο φως, και αντίστοιχα συλλέκτες (IR Receiver), η οποίοι δίνουν την δυνατότητα εντοπισμού πραγμάτων τα οποία δεν είναι δυνατόν να δει ένα ανθρώπινο μάτι.

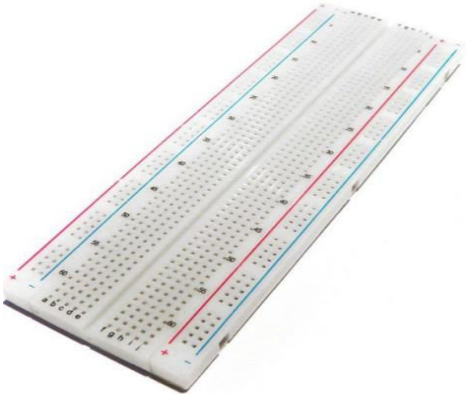


Εικόνα 4.12 **[29]**

Στην εικόνα 4.12 στο σχήμα (α) φαίνεται η εκπομπή υπέρυθρου φωτός από ένα LED σε ένα ορισμένο εύρος, το οποίο δεν φαίνεται να εντοπίζει κάποιο αντικείμενο. Στο σχήμα (β) βλέπουμε ξανά την εκπομπή υπέρυθρου φωτός από ένα LED σε ένα ορισμένο εύρος, το οποίο όταν φτάσει πάνω σε κάποια επιφάνεια, τότε το αντικείμενο δημιουργεί μια αντανάκλαση και το υπέρυθρο φως φτάνει πίσω, δίνοντας την δυνατότητα στον αισθητήρα ή αλλιώς ο συλλέκτης υπέρυθρου φωτός, να μπορεί να εντοπίζει τα αντικείμενα και επομένως να τα αναπαριστά.

8. Πειραματική Διάταξη 830 οπών (Breadboard 830 Points)

Χρησιμοποιήθηκε έτσι ώστε να μπορέσουμε να διασυνδέουμε σε αυτό, όλα τα απαραίτητα εξαρτήματα που χρειαζόμασταν για την δημιουργία του έργου μας (Εικόνα 4.13).



Εικόνα 4.13 Πειραματική Διάταξη 830 οπών

9. GPIO Πίνακας Επέκτασης 40 ακροδεκτών (GPIO Extension Board 40 pin)

Ο Πίνακας επέκτασης, χρησιμοποιήθηκε έτσι ώστε να επεκτείνουμε τους 40 ακροδέκτες που βρίσκονται στο Raspberry Pi, και να τους <<μεταφέρουμε>> πάνω στην πειραματική διάταξη. Αυτό πραγματοποιήθηκε, έτσι ώστε να μην συνδέουμε τα εξαρτήματα μας απευθείας πάνω στους ακροδέκτες του Raspberry Pi, ώστε να μην προκαλέσουμε κάποια ζημιά και τέλος να μπορέσουμε να διασυνδέσουμε τα εξαρτήματα πάνω στην πειραματική διάταξη με την σειρά την οποία επιθυμούμε (Εικόνα 4.14).

Εικόνα 4.14 GPIO Πίνακας Επέκτασης και καλώδιο διασύνδεσης

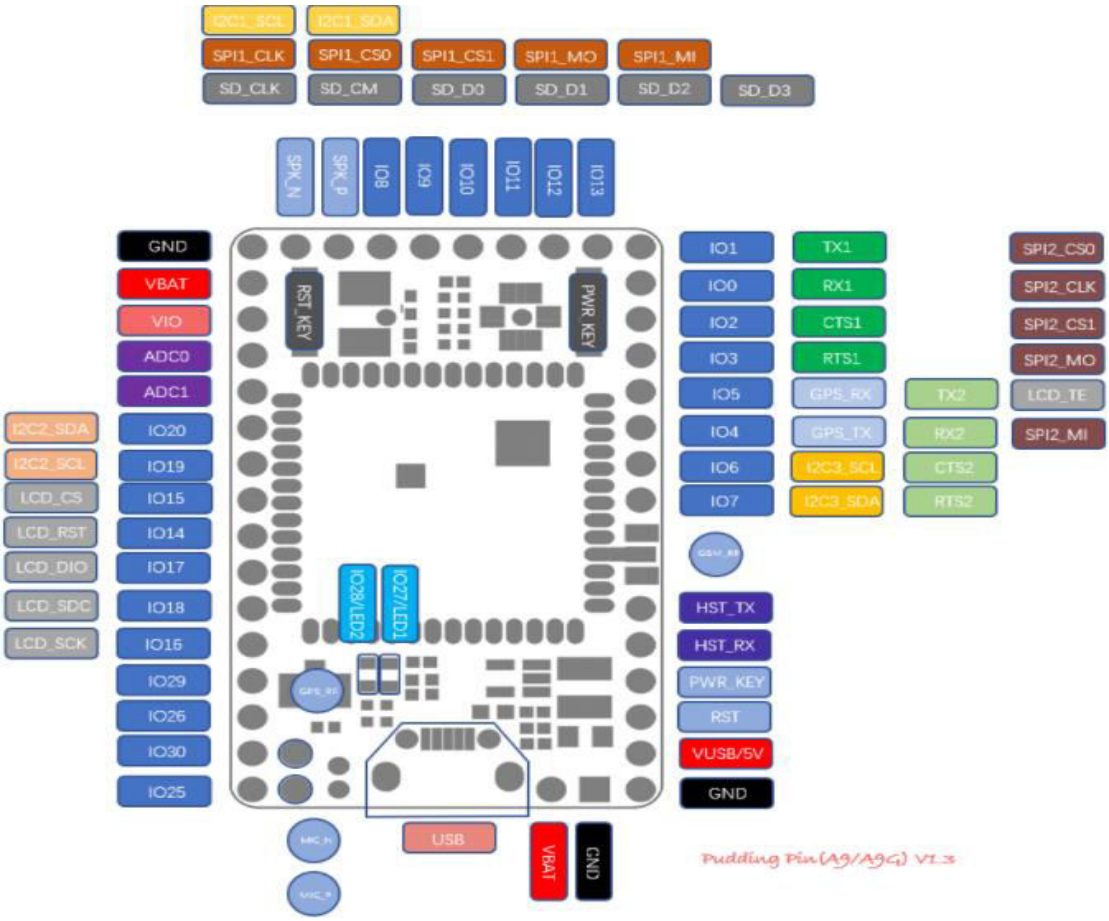
10. A9G GSM/GPRS+GPS Module

Το A9G GSM/GPRS+GPS Module δημιουργήθηκε από την εταιρία Ai-Thinker, η οποία εταιρία προσφέρει ένα μεγάλο εύρος προϊόντων και λύσεων πάνω στον χώρο του IoT. [30]

Το A9G GSM/GPRS+GPS Module μας δίνει την δυνατότητα να εντοπίσουμε την θέση του εκάστοτε οδηγού στο σύστημα μας, καθώς περιέχει GPS και ακόμα να διαχειριστούμε υπηρεσίες τηλεπικοινωνιών, όπως εξυπηρέτηση φωνητικών υπηρεσιών και μετάδοση δεδομένων με τη χρήση της ψηφιακής διαμόρφωσης, καθώς περιέχει GSM/GPRS μονάδα (Εικόνα 4.15). Η χρήση του GPS και της GSM/GPRS μονάδας γίνεται μέσα από ένα μεγάλο πλήθος AT εντολών. [31]

Εικόνα 4.15 A9G GSM/GPRS+GPS Module

Ένα ακόμη σημαντικό χαρακτηριστικό του A9G GSM/GPRS+GPS Module, είναι ότι αποτελείται από 29 GPIO Pins τα οποία μπορούμε να διαχειριστούμε και μέσα από το λογισμικό (Εικόνα 4.16).



Εικόνα 4.16 26 GPIO Pins A9G GSM/GPRS+GPS Module **[32]**

Ας δούμε λοιπόν, κάποια από τα Pin που χρησιμοποιήθηκαν και στο έργο μας. Αρχικά από τα δύο κάτω δεξιά, το ένα αφορά την γείωση (GND) και το άλλο την τροφοδοσία (VUSB/5V) του A9G GSM/GPRS+GPS Module. Βέβαια, ένα άλλος τρόπος για να τροφοδοτήσουμε το A9G Module είναι μέσω USB διεπαφής (USB). Ας δούμε τα δύο ακριβώς από πάνω Pin (RST & PWR_KEY), το RST χρησιμοποιείται για την επαναφορά του A9G, και το PWR_KEY για την ενεργοποίηση του. Τα άλλα δύο Pin που έχουνε χρησιμοποιηθεί στο έργο, βρίσκονται πάνω δεξιά. Το πρώτο Pin αφορά τον πομπό (TX1) και χρησιμοποιείται για την μετάδοση των δεδομένων, και το δεύτερο Pin το οποίο βρίσκεται ακριβώς από κάτω, αφορά τον δέκτη (RX1) και χρησιμοποιείται για την λήψη των δεδομένων.

Χαρακτηριστικά A9G GSM/GPRS+GPS Module:

- Διεπαφή μετατροπής αναλογικού σήματος σε ψηφιακό (ADC)
- Διαχείριση φόρτισης μπαταριών λιθίου
- Υποδοχή micro SIM κάρτας

- Υποδοχή microSD κάρτας
- 1 button ενεργοποίησης
- Σειριακή διεπαφή (SPI)
- 1 button επαναφοράς
- I2C2 διεπαφή
- Διεπαφή USB
- Μικρόφωνο

11. 3 Δίοδοι φωτοεκπομπής (LED)(1 Πράσινο, 1 Κίτρινο, 1 Κόκκινο)

Χρησιμοποιήθηκαν ώστε να ενεργοποιούνται ανάλογα με την κατάσταση του που βρίσκεται ο οδηγός (Εικόνα 4.17).

Πράσινο LED = Καλή κατάσταση του οδηγού

Κίτρινο LED = Κάθε φορά που υπάρχει προειδοποίηση

Κόκκινο LED = Όταν ο οδηγός έχει ξεπεράσει το όριο των προειδοποιήσεων

Εικόνα 4.17 Κόκκινος, Κίτρινος και Πράσινος δίοδος φωτοεκπομπής

12. Αντίσταση 220Ω

Η αντίσταση χρησιμοποιείται, ώστε να μείωση στη διέλευση του ηλεκτρικού ρεύματος, και στην περίπτωσης μας επιλέχθηκαν αντιστάσεις των 220Ω ώστε να μειώσουν το ρεύμα που θα διαρρέει τις διόδους φωτοεκπομπής (Εικόνα 4.18).

Εικόνα 4.18 Αντίσταση 220Ω

13. **Καλώδιο Βραχυκυκλωτήρα αρσενικό σε αρσενικό (Jumper Wire M-M) (Εικόνα 4.19)**

Εικόνα 4.19 Καλώδιο βραχυκυκλωτήρα αρσενικό σε αρσενικό

- **Καλώδιο Βραχυκυκλωτήρα αρσενικό σε θηλυκό (Jumper Wire M-F) (Εικόνα 4.20)**

Εικόνα 4.20 Καλώδιο βραχυκυκλωτήρα αρσενικό σε θηλυκό

4.4 Συναρμολόγηση Υλικού

Σε αυτήν την ενότητα, θα δούμε με βήμα-βήμα πως μπορούμε να συναρμολογήσουμε όλο το απαραίτητο υλικό, που αναφέραμε αναλυτικά στην ενότητα 3.3, έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε το Raspberry Pi, με όλο το απαραίτητο εξωτερικό υλικό, και έπειτα να συνεχίσουμε στην εγκατάσταση του απαραίτητου λογισμικού.

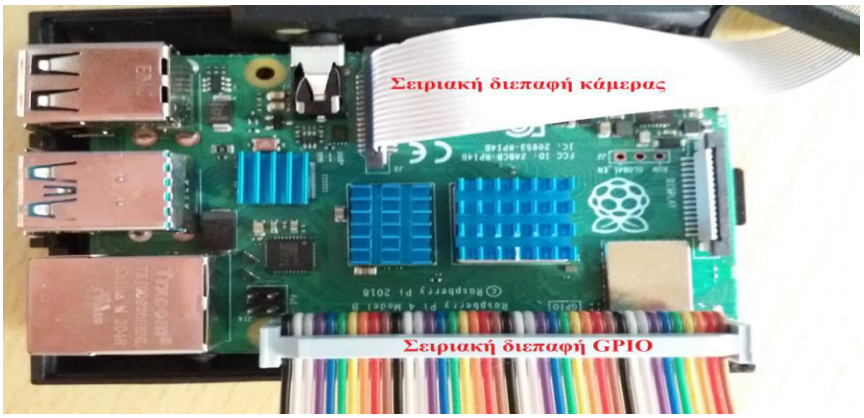
Βήμα 1^{ο}

Αρχικά θα πρέπει να τοποθετήσουμε την κάρτα microSD στο πίσω μέρος του Raspberry Pi (Εικόνα 4.21)

Εικόνα 4.21 Τοποθέτηση microSD κάρτα στο Raspberry Pi **[33]**

Βήμα 2ο

Τοποθετούμε το κάτω μέρος της θήκης κάτω από το Raspberry Pi, ύστερα συνδέουμε την σειριακή διεπαφή της κάμερας στο Raspberry Pi, με τον ίδιο τρόπο συνδέουμε την σειριακή διεπαφή του GPIO πίνακα επέκτασης (Εικόνα 4.22).



Εικόνα 4.22 Σειριακή διεπαφή Κάμερας και GPIO πίνακα επέκτασης

Βήμα 3ο

Περνάμε την σειριακή διεπαφή της κάμερας μέσα από το επάνω μέσω της θήκης ώστε να βγει προς τα έξω, με τον ίδιο τρόπο περνάμε προς τα έξω την σειριακή διεπαφή του GPIO. Τέλος κλείνουμε την θήκη και συνδέουμε την κάμερα στην σειριακή διεπαφή (Εικόνα 4.23).

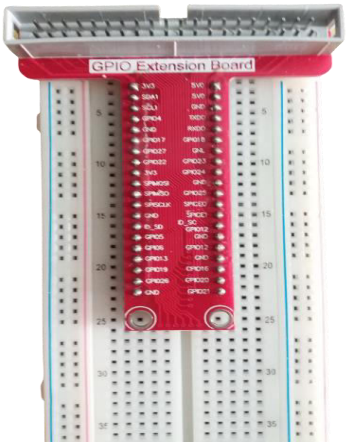
Εικόνα 4.23 Κάμερα συνδεδεμένη με την σειριακή διεπαφή κάμερας

Βήμα 4ο

Σε αυτό το βήμα θα συνδέσουμε μια συσκευή εξόδου ήχου. Ο πρώτος τρόπος είναι να συνδέσουμε οποιαδήποτε ηχεία/ακουστικά που έχουμε και δύνανται να συνδεθούν στην θύρα 'Audio + Video' του Raspberry Pi, όπου δέχεται ένα κλασσικό καλώδιο τύπου 'jack 3.5mm' (βλέπε υποενότητα 3.3 Αναλυτικά το Υλικό → Εικόνα 4.3 → 'Audio+Video' Composite 3.5mm Jack). Ο δεύτερος τρόπος είναι να συνδέσουμε Bluetooth ηχεία στο Raspberry Pi, τα οποία δεν απαιτούν την χρήση καλωδίου για την σύνδεση τους στο Raspberry Pi, απλά για να συνδεθούμε θα πρέπει να ενεργοποιήσουμε τα Bluetooth του ηχείου, ύστερα να ανοίξουμε τα Bluetooth του Raspberry Pi, έπειτα να βρούμε το αναγραφόμενο όνομα των ηχείων και τέλος να πραγματοποιήσουμε την σύζευξη μεταξύ τους.

Βήμα 5ο

Τοποθετούμε τον GPIO πίνακας επέκτασης πάνω στην πειραματική διάταξη (Εικόνα 4.24).



Εικόνα 4.24 GPIO πίνακα επέκτασης τοποθετημένος στην πειραματική διάταξη

Βήμα 6ο

Συνδέουμε την σειριακή διεπαφή του GPIO πάνω στον GPIO πίνακα επέκτασης (Εικόνα 4.25).

Εικόνα 4.25 GPIO πίνακα επέκτασης συνδεδεμένος με σειριακή διεπαφή GPIO

Βήμα 7^{ο}

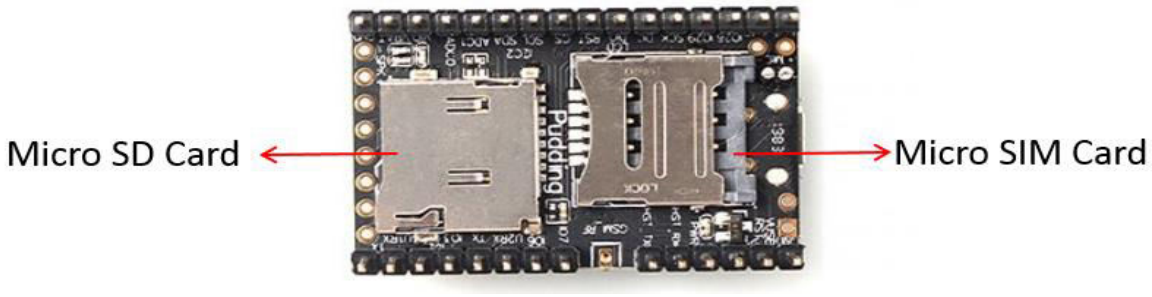
Επόμενο βήμα, είναι να συνδέσουμε το ανεμιστηράκι πάνω στον πίνακα διάταξης, χρησιμοποιώντας 2 καλώδια βραχυκυκλωτήρα αρσενικό σε αρσενικό, αρχικά συνδέουμε το 1^{ο} άκρο πάνω στην θετική υποδοχή του ανεμιστήρα (μαύρο καλώδιο) και έπειτα συνδέουμε το 2^{ο} άκρο πάνω στην αρνητική υποδοχή του ανεμιστήρα (κόκκινο καλώδιο), τέλος συνδέουμε το κόκκινο καλώδιο στα 5V και το μαύρο καλώδιο στην γείωση (Εικόνα 4.26).

Εικόνα 4.26 Σύνδεση θετικού και αρνητικού πόλου ανεμιστήρα στην πειραματική διάταξη

Βήμα 8^{ο}

Σε αυτό το βήμα θα πρέπει να τοποθετήσουμε την micro SIM κάρτα στο A9G GSM/GPRS+GPS Module. Αρχικά στο πίσω μέρος του A9G Module βλέπουμε 2 υποδοχές, η μια αφορά την υποδοχή κάρτας micro SIM (Micro SIM Card), και η άλλη την

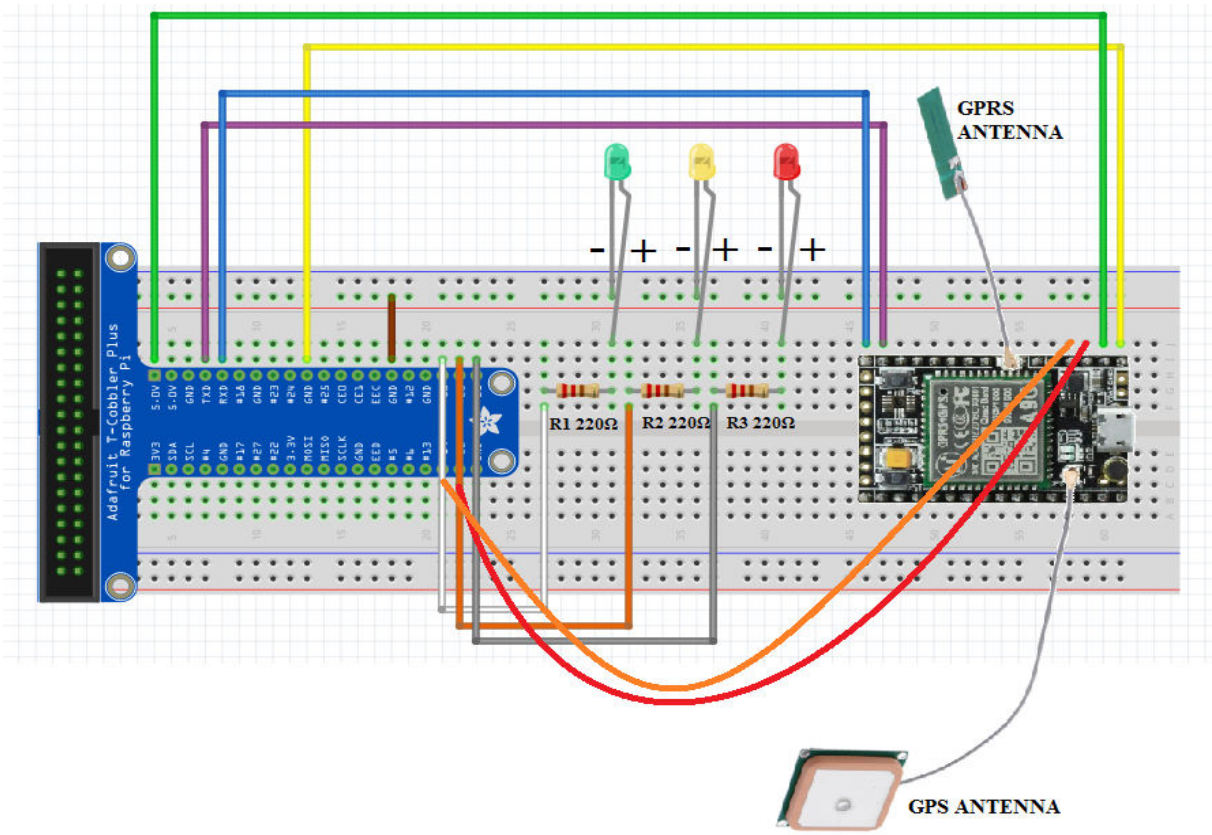
υποδοχή micro SD κάρτας (Micro SD Card) (Εικόνα 4.27). Επομένως εμείς τοποθετούμε την micro SIM κάρτα μας στην αντίστοιχη υποδοχή.



Εικόνα 4.27 Πίσω όψη A9G GSM/GPRS+GPS Module

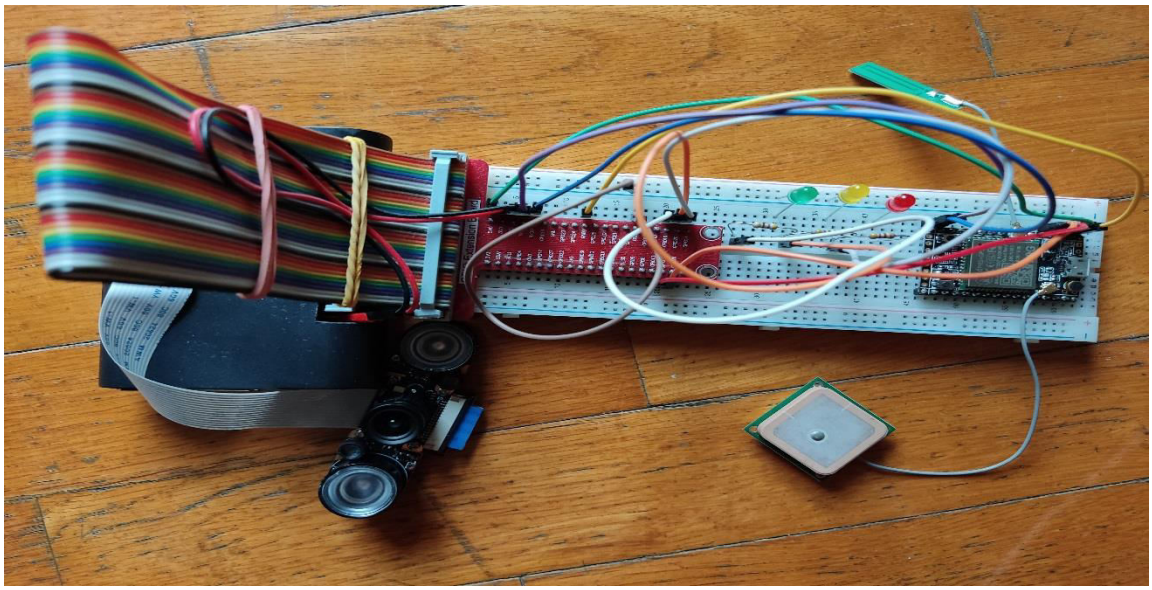
Βήμα 9ο

Τέλος τοποθετούμε στον πίνακα διάταξης τις διόδους φωτοεκπομπής, τις αντιστάσεις και το A9G GSM/GPRS+GPS Module (μαζί με τις κεραίες GPS & GPRS) όπως βλέπουμε στην παρακάτω διάταξη (Εικόνα 4.28). Η σχεδίαση υλοποιήθηκε με το πρόγραμμα Fritzing (https://fritzing.org/) εκτός από το A9G GSM/GPRS+GPS Module (μαζί με τις κεραίες) το όποιο δημιουργήθηκε και προστέθηκε από πρόγραμμα επεξεργασίας εικόνας.



Εικόνα 4.28 Γραφική αναπαράσταση της πραγματικής πειραματικής διάταξης

Ακόμα βλέπουμε την πλήρη διάταξη του υλικού υλοποιημένη στο σύνολο της (Εικόνα 4.29) όπως αναλύσαμε παραπάνω.



Εικόνα 4.29 Πραγματική υλοποίηση του υλικού

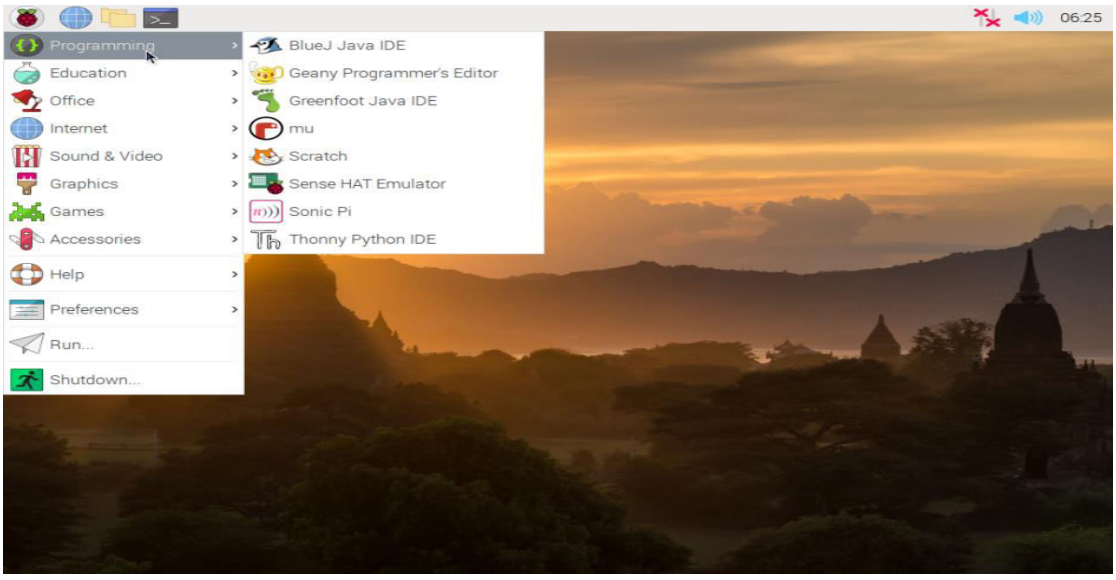
5. Περιγραφή Λογισμικού

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα κάνουμε μια αναφορά στο λειτουργικό σύστημα Raspberry Pi OS. Έπειτα θα περιγράψουμε την γλώσσα Python την οποία χρησιμοποιήσαμε για την υλοποίηση του έργου μας. Ακόμα θα δούμε όλες τις απαραίτητες βιβλιοθήκες, τα αρθρώματα (modules) και όλα τα βασικά στοιχεία που χρησιμοποιήθηκαν στο έργο μας.

5.2 Raspberry Pi OS

Το Raspberry Pi OS (πρώην Raspbian), είναι ένα λειτουργικό σύστημα το οποίο βασίζεται στον Debian. Από το 2015 παρέχεται επίσημα από το Raspberry Pi Foundation, ώς το κύριο λειτουργικό σύστημα για το Raspberry Pi. Η διαφορά του με άλλες διανομές Linux, είναι ότι το Raspberry Pi OS είναι βελτιστοποιημένο για την εκτέλεση σε επεξεργαστές ARM και συγκεκριμένα διαμορφωμένο για τα Raspberry Pi. Χρησιμοποιεί μια τροποποιημένη έκδοση του LXDE ώς επιφάνεια εργασίας και επιπλέον χρησιμοποιεί τον διαχειριστή παραθύρων Openbox (Εικόνα 5.1). Έρχεται με την γλώσσα Python προεγκατεστημένη στο σύστημα, όπως και με διάφορα προγραμματιστικά εργαλεία και προγράμματα. Η εγκατάσταση των πακέτων γίνεται μέσω του εργαλείου APT. Τέλος διαθέτει 3 επιλογές εγκατάστασης (Lite, Κανονική & Πλήρης). [34]



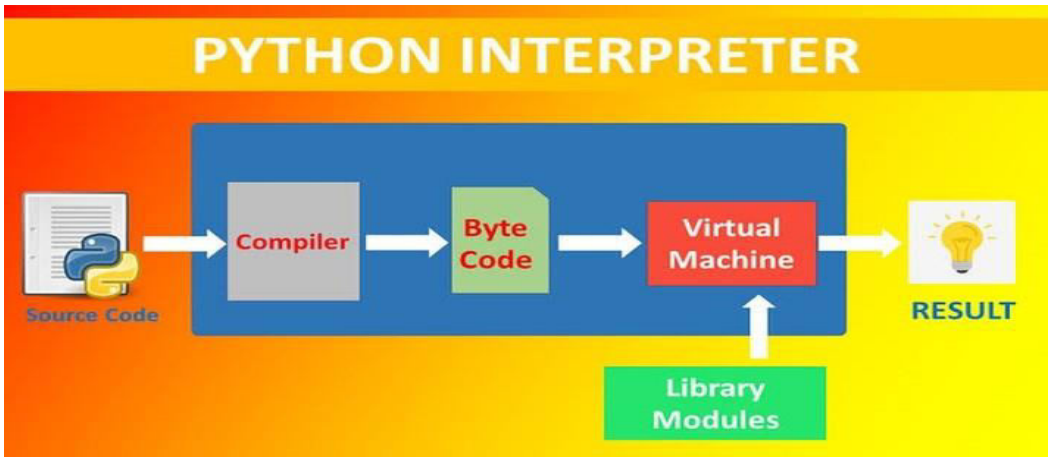
Εικόνα 5.1 Επιφάνεια εργασίας Raspberry Pi OS

5.3 Γλώσσα Python

Η Python είναι μια ισχυρή γλώσσα προγραμματισμού, με απλό συντακτικό και εξαιρετική αναγνωσιμότητα. Οι κύριοι στόχοι του σχεδιασμού της Python, είναι ο κώδικας της να είναι ευανάγνωστος για κάποιον που θέλει να το κατανοήσει, όπως και να μπορεί να συντάσσεται με απλό και κατανοητό τρόπο.

Η Python πρωτοεμφανίστηκε το 1991 με τον δημιουργό της να είναι ο Guido van Rossum, ο οποίος είχε ξεκινήσει να υλοποιεί την Python από το 1989 εργαζόμενος στο CWI (Εθνικό Ινστιτούτο Ερευνών στην Ολλανδία) ως διάδοχος της γλώσσας προγραμματισμού ABC.

Η Python είναι μια διερμηνευόμενη (interpreted) γλώσσα προγραμματισμού. Αυτό σημαίνει ότι το πρόγραμμα που έχουμε γράψει, μεταφράζεται από τον διερμηνευτή της Python, δηλαδή μεταγλωττίζει γραμμή-γραμμή το πρόγραμμα σε μορφή bytes (.pyc ή .pyo), ώστε αυτό έπειτα να μπορέσει να εκτελεστεί από τον διερμηνέα της Python (γνωστός ως Python Virtual Machine)(Εικόνα 5.2). Η διαφορά της Python με γλώσσες όπως C, C++ είναι ότι αυτές μεταγλωττίζονται σε γλώσσα μηχανής και εκτελούνται απευθείας στον επεξεργαστή. Συνεπώς αυτό κάνει τις γλώσσες C, C++ και πιο γρήγορες στην εκτέλεση τους.



Εικόνα 5.2 Απεικόνιση βημάτων του διερμηνευτή της Python **[35]**

Ακόμα η Python υποστηρίζει μοντέλα διαδικαστικού και αντικειμενοστραφές προγραμματισμού. Είναι δυναμική γλώσσα και χρησιμοποιεί την συλλογή απορριμμάτων (garbage collection), η οποία είναι μια μορφή αυτόματης διαχείρισης μνήμης που λειτουργεί στο υπόβαθρο κατά την διάρκεια της εκτέλεσης του

προγράμματος, και ο σκοπός της είναι να διαγράφει αντικείμενα, τα οποία δεν χρησιμοποιούνται πλέον από το πρόγραμμα, με σκοπό την απελευθέρωση την μνήμης.

5.4 Βιβλιοθήκες που χρησιμοποιήθηκαν στο έργο

Παρακάτω θα περιγράψουμε και θα εξηγήσουμε την χρησιμότητα όλων των βιβλιοθηκών και αρθρωμάτων (modules) που εισάγαμε στο έργο μας.

5.4.1 Λίστα βιβλιοθηκών και αρθωμάτων (modules)

1. OpenCV (cv2)
2. imutils
3. dlib
4. numpy
5. threading
6. Rpi.GPIO
7. time
8. random
9. requests
10. smtplib
11. smtplib
12. email
13. os
14. serial

5.4.2 Περιγραφή βιβλιοθηκών και αρθρωμάτων (modules)

1. **OpenCV (cv2)**

   Η OpenCV βιβλιοθήκη περιέχει ένα σύνολο συναρτήσεων, η οποίες εφαρμόζονται στην μηχανική όραση, σε εφαρμογές που απαιτούν την ανάλυση των αντικειμένων τους σε πραγματικό χρόνο, αλλά και στην επεξεργασία εικόνας όπου η ανάλυση των αντικειμένων πραγματοποιείται σε μη πραγματικό χρόνο.

   Είναι μια βιβλιοθήκη η οποία αναπτύχθηκε από την Intel τον Ιούνιο του 2000 και στην συνέχεια υποστηρίχθηκε από την WillowGarage. Η ανάπτυξη της βιβλιοθήκης

ξεκίνησε με σκοπό την δημιουργία προηγμένων εφαρμογών πάνω στην μηχανική όραση, καθώς και στην δημιουργία 3D απεικονίσεων.

Η βιβλιοθήκη OpenCV είναι γραμμένη σε C/C++ αλλά διαθέτει διεπαφές οι οποίες δίνουν την δυνατότητα σε γλώσσες όπως Python, Java, MATLAB/OCTAVE να μπορούν να χρησιμοποιούν την OpenCV βιβλιοθήκη. [36]

2. **imutils**

Η βιβλιοθήκη imutils αναπτύχθηκε αρχικά από τον Adrian Rosebrock, με εφαρμογή την επεξεργασία εικόνας, και στόχος της είναι να προσφέρει ένα εύρος βασικών λειτουργιών που εφαρμόζονται στην επεξεργασία εικόνας, όπως περιστροφή, αλλαγή μεγέθους εικόνας, σκελετοποίηση κτλ. Οι συναρτήσεις που περιέχει η βιβλιοθήκη είναι εύκολες στην χρήση τους και έχουν άμεση διεπαφή με την βιβλιοθήκη OpenCV. Επίσης η βιβλιοθήκη imutils είναι γραμμένη σε γλώσσα Python. [37]

3. **dlib**

Η βιβλιοθήκη dlib περιέχει ένα μεγάλο εύρος αλγορίθμων και εργαλείων μηχανικής μάθησης οι οποίοι εφαρμόζονται για την δημιουργία λογισμικού και την επίλυση προβλημάτων σε τομείς όπως ρομποτική, όραση υπολογιστών, δίκτυα υπολογιστών, ενσωματωμένα συστήματα κτλ. Η βιβλιοθήκη dlib είναι γραμμένη σε C++, αλλά διαθέτει διεπαφή η οποία δίνει τη δυνατότητα πρόσβασης και στη γλώσσα Python. [17]

4. **mediapipe**

Η MediaPipe αποτελεί ένα πλαίσιο λογισμικού (framework) το όποιο δίνει την δυνατότητα ανάπτυξης πολύπλοκών εφαρμογών, οι οποίες έχουν ανάγκη να χρησιμοποιήσουν πολύπλοκούς αλγορίθμους μηχανικής μάθησης και μηχανικής όρασης. Προσφέρει ένα μεγάλο εύρος αλγορίθμων κάποιοι από τους οποίους δίνουν την δυνατότητα εντοπισμού προσώπου, εντοπισμού χεριών, εντοπισμού διαφόρων χαρακτηριστικών σώματος, εντοπισμού αντικειμένου κτλ. Συνήθως χρησιμοποιείται σε εφαρμογές που απαιτούν την χρήση του σε πραγματικό χρόνο, καθώς μπορεί να αξιοποιήσει αποτελεσματικά τους εκάστοτε πόρους (CPU ή GPU) του συστήματος όπου χρησιμοποιείται, δίνοντας με αυτόν τον τρόπο ένα πολύ αξιόπιστό και γρήγορο αποτέλεσμα.

Η MediaPipe αναπτύχθηκε και πρωτοεμφανίστηκε αρχικά από την Google τον Ιούνιου του 2019 και αποτελεί ένα λογισμικό ανοιχτού κώδικα, στο οποίο ο καθένας μπορεί να έχει πρόσβαση.

Η MediaPipe είναι γραμμένη κυρίως στην γλώσσα C++, αλλά δίνεται η δυνατότητα χρήση της, μέσω διεπαφών σε γλώσσες όπως Python, Java , JavaScript και Objective-C. [38]

5. **numpy**

   Η βιβλιοθήκη numpy δημιουργήθηκε από τον Travis Oliphant το 2005, και αποτελεί βασική υποδομή για επιστημονικές εφαρμογές. Υποστηρίζει τη δημιουργία μεγάλου εύρους πολυδιάστατων πινάκων και συστοιχιών, δίνοντας τη δυνατότητα εκτέλεσης υψηλών μαθηματικών συναρτήσεων πάνω σε αυτές. Το μεγαλύτερο κομμάτι της βιβλιοθήκης είναι γραμμένο την γλώσσα Python αλλά τα κομμάτια που απαιτούν μεγαλύτερη ταχύτητα εκτέλεσης είναι γραμμένα σε γλώσσες C/C++. [39]

6. **threading**

   Είναι ένα άρθρωμα (module) το οποίο βρίσκεται ενσωματωμένο στη πρότυπη βιβλιοθήκη της Python. Χρησιμοποιείται έτσι ώστε να μπορούμε να τρέχουμε πολλά διαφορετικά νήματα ταυτόχρονα και να μπορούμε να τα διαχειριζόμαστε. Ο κύριος σκοπός τους είναι να μπορούμε να τρέχουμε διαφορετικές διεργασίες και συναρτήσεις την ίδια χρονική στιγμή αλλά σε διαφορετικά νήματα.

7. **Rpi.GPIO**

   Η Βιβλιοθήκη Rpi.GPIO δίνει την δυνατότητα πρόσβασης και διαχείρισης των 40 GPIO pins που βρίσκονται πάνω στο Raspberry Pi.

8. **time**

   Είναι ένα άρθρωμα(module) το οποίο βρίσκεται ενσωματωμένο στη πρότυπη βιβλιοθήκη της Python. Δίνει τη δυνατότητα χρήσης συναρτήσεων οι οποίες συσχετίζονται με την ώρα και τον χρόνο.

9. **random**

   Είναι ένα άρθρωμα το οποίο βρίσκεται ενσωματωμένο στη πρότυπη βιβλιοθήκη της Python. Περιέχει συναρτήσεις με γεννήτριες ψευδοτυχαίων αριθμών και μας δίνει την δυνατότητα να παράγουμε «ψευδοτυχαίους» αριθμούς με διαφορετικούς τρόπους.

10. **requests**

Είναι μια βιβλιοθήκη η οποία αναπτύχθηκε από τον Kenneth Reitz το 2011 και μας δίνει την δυνατότητα να ανταλλάσσουμε HTTP αιτήματα μέσω υπηρεσιών ιστού. Παρέχει ένα μεγάλο εύρος συναρτήσεων που μας δίνουν την δυνατότητα επικοινωνίας με υπηρεσίες ιστού μέσω HTTP αιτημάτων, με έναν πιο εύκολο τρόπο.

11. smtplib

Είναι ένα άρθρωμα το οποίο βρίσκεται ενσωματωμένο στην πρότυπη βιβλιοθήκη της Python. Αφορά την εγκαθίδρυση μιας SMTP (Simple Mail Transfer Protocol) συνεδρίας πελάτη, δίνοντας την δυνατότητα αποστολής ηλεκτρονικής αλληλογραφίας μέσω κάποιου SMTP διακομιστή.

12. email

Είναι ένα άρθρωμα το οποίο βρίσκεται ενσωματωμένο στην πρότυπη βιβλιοθήκη της Python. Προσφέρει συναρτήσεις οι οποίες μας δίνουν την δυνατότητα δημιουργίας ηλεκτρονικών μηνυμάτων αλληλογραφίας, με σωστά δομημένο τρόπο, έτσι ώστε να είναι συμβατά με τον εκάστοτε διακομιστή ηλεκτρονικού ταχυδρομείου.

13. os

Είναι ένα άρθρωμα το οποίο βρίσκεται ενσωματωμένο στην πρότυπη βιβλιοθήκη της Python και παρέχει λειτουργίες ανάλογα με το εκάστοτε λειτουργικό σύστημα που χρησιμοποιείται. Παρέχει αλληλεπίδραση με το λειτουργικό σύστημα μέσω συναρτήσεων οι οποίες π.χ. μπορούν να δημιουργήσουν, να διαγράψουν έναν κατάλογο, να εκτελέσουν CMD ή Bash εντολές και γενικά να εκτελέσουν διεργασίες του λειτουργικού συστήματος με έναν πιο ευέλικτο τρόπο.

14. serial

Είναι ένα άρθρωμα το οποίο βρίσκεται ενσωματωμένο στην πρότυπη βιβλιοθήκη της Python. Χρησιμοποιείται για την πρόσβαση και τη επικοινωνία με σειριακές πόρτες σε διάφορα λειτουργικά συστήματα όπως π.χ. Windows, OSX (Mac), Linux, BSD κτλ.

6. Εγκατάσταση Λογισμικού

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα δούμε πως μπορούμε να εγκαταστήσουμε το λειτουργικό σύστημα Raspberry Pi OS, το οποίο χρησιμοποιήθηκε ως το κύριο λειτουργικό για το Raspberry Pi. Έπειτα θα δημιουργήσουμε ένα εικονικό περιβάλλον στο οποίο θα πραγματοποιήσουμε όλες τις απαραίτητες εγκαταστάσεις, χωρίς να επηρεάσουμε τις ήδη υπάρχουσες εγκαταστάσεις τους συστήματος. Ακόμα θα δούμε βήμα-βήμα την εγκατάσταση των απαραίτητων βιβλιοθηκών και αρθρωμάτων (modules), που ήταν απαραίτητα έτσι ώστε να μπορέσουμε να τρέξουμε το πρόγραμμα μας. Τέλος θα εγκαταστήσουμε όλα τα απαραίτητα πακέτα που χρειαστήκαμε για την εκτέλεση του προγράμματος μας.

6.2 Εγκατάσταση Raspberry Pi OS

Βήμα 1^{ο}

Αρχικά μεταβαίνουμε στην επίσημη ιστοσελίδα του Raspberry Pi (https://www.raspberrypi.org/software/operating-systems/) όπου μπορούμε να βρούμε το επίσημο λειτουργικό σύστημα Raspberry Pi OS το οποίο είναι συμβατό με όλες τις εκδόσεις του Raspberry Pi και κατεβάζουμε την προτεινόμενη έκδοση **'Raspberry Pi OS with desktop and recommended software'** όπως βλέπουμε στην Εικόνα 6.1.

Εικόνα 6.1 Επίσημη Ιστοσελίδα του Raspberry Pi

Βήμα 2ο

Έπειτα θα κατεβάσουμε και θα χρησιμοποιήσουμε το λογισμικό balenaEtcher έτσι ώστε να περάσουμε το λειτουργικό σύστημα 'Raspberry Pi OS' στην microSD κάρτα μας. Επομένως μεταβαίνουμε στην επίσημη ιστοσελίδα του λογισμικού (https://www.balena.io/etcher/) και κατεβάζουμε το λογισμικό όπως όπως βλέπουμε στην Εικόνα 6.2 (Η διαδικασία που ακολουθεί αφορά χρήστες Windows, μπορούμε να κατεβάσουμε και να εγκαταστήσουμε το λογισμικό και σε διαφορετικό λειτουργικό σύστημα (Linux ή macOS).

Εικόνα 6.2 Επίσημη Ιστοσελίδα του λογισμικού balenaEtcher

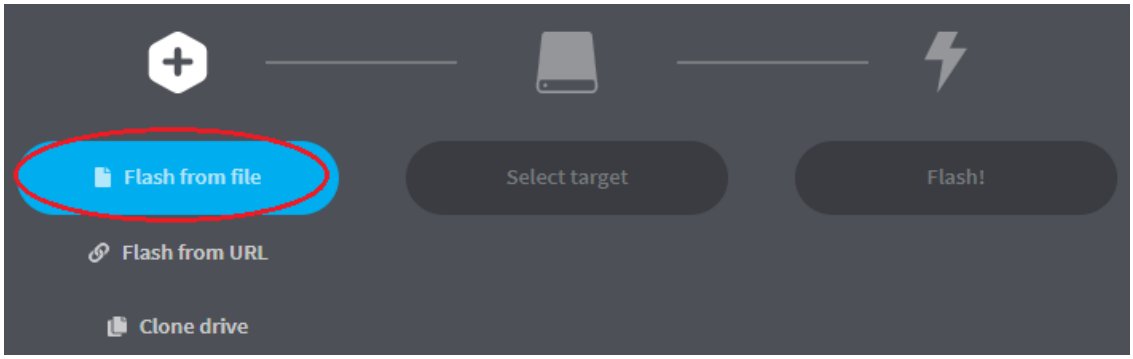
Βήμα 3ο

Συνεχίζοντας εισάγουμε την microSD κάρτα στον υπολογιστή μας ώστε να μπορέσουμε να περάσουμε το λειτουργικό σύστημα. Ένας τρόπος για να το πραγματοποιήσουμε αυτό είναι να χρησιμοποιήσουμε πιθανόν κάποιον αντάπτορα που θα δέχεται microSD κάρτα και θα συνδέεται με USB (microSD to USB Card Reader) όπως βλέπουμε στην Εικόνα 6.3. Βέβαια οποιοσδήποτε διαφορετικός τρόπος για να συνδέσουμε την microSD κάρτα μας στον υπολογιστή μας είναι αποδεκτός.

Εικόνα 6.3 Αντάπτορας microSD κάρτας σε USB

Βήμα 4ο

Έπειτα ανοίγουμε και εγκαθιστούμε το λογισμικό balenaEtcher που κατεβάσαμε στο **Βήμα 2.** Μόλις ολοκληρωθεί η εγκατάσταση πατάμε στην επιλογή **'Flash from file'** (Εικόνα 6.4), εκεί θα πρέπει να επιλέξουμε το αρχείο του λειτουργικού συστήματος που κατεβάσαμε στο **Βήμα 1**.



Εικόνα 6.4 Επιλογή αρχείου λειτουργικού συστήματος από το λογισμικό balenaEtcher

Βήμα 5ο

Μετά επιλέγουμε **'Select target'** και διαλέγουμε την microSD κάρτα που έχουμε συνδέσει στον υπολογιστή μας (Εικόνα 6.5).

Εικόνα 6.5 Επιλογή υλικού εγγραφής από το λογισμικό balenaEtcher

Βήμα 6ο

Τέλος επιλέγουμε **'Flash!'** περιμένουμε λίγη ώρα μέχρι να πραγματοποιηθεί η διαδικασία εγγραφής στη κάρτα, μόλις η διαδικασία τελειώσει μπορούμε να τοποθετήσουμε την microSD κάρτα στο Raspberry Pi (τοποθετούμε την κάρτα όπως έχουμε δει στην ενότητα 4.4 Συναρμολόγηση Υλικού → Βήμα 1ο) και να ενεργοποιήσουμε το Raspberry Pi. Πλέον είμαστε έτοιμη να χρησιμοποιήσουμε το λειτουργικό σύστημα Raspberry Pi OS.

6.3 Δημιουργία Εικονικού Περιβάλλοντος (Virtual Environment)

6.3.1 Ορισμός Εικονικού Περιβάλλοντος

Ένα Εικονικό Περιβάλλον, μας δίνει την δυνατότητα δημιουργίας ενός ξεχωριστού απομονωμένου περιβάλλοντος από αυτόν του συστήματος, έτσι ώστε να μπορούμε να εγκαθιστούμε πακέτα και βιβλιοθήκες οι οποίες θα μας είναι χρήσιμες μόνο για ένα project κάθε φορά. Κάποια από τα πλεονεκτήματα που μας δίνουν τα εικονικά περιβάλλοντα είναι:

- Απομόνωση όλων των εγκαταστάσεων και των προγραμμάτων από το υπόλοιπο σύστημα, έτσι ώστε γίνεται χρήση τους, μόνο όταν εμείς το χρειαζόμαστε.
- Χρήση διαφορετικών εκδόσεων μιας βιβλιοθήκης.
- Χρήση βιβλιοθήκης η οποία δεν θα χρησιμοποιηθεί από αλλού.
- Ευελιξία σε δοκιμές και εγκαταστάσεις χωρίς τον φόβο να κάνουμε κάποιο λάθος.
- Εύκολη δημιουργία/διαγραφή εικονικού περιβάλλοντος.

Με λίγα λόγια όταν θέλουμε να ξεκινήσουμε ένα νέο project τότε καλό είναι να δημιουργούμε ένα ξεχωριστό εικονικό περιβάλλον όπου θα συμπεριλαμβάνουμε ότι εγκαταστάσεις χρειαζόμαστε συγκεκραμένα για το project μας, χωρίς να εμπλεκόμαστε με τις υπόλοιπές εγκαταστάσεις του συστήματος.

6.3.2 Εγκατάσταση Εικονικού Περιβάλλοντος

Βήμα 1ο

Εγκατάσταση του πακέτου pip που χρησιμοποιείται έτσι ώστε να μας δίνει την δυνατότητα εγκαταστάσεων πακέτων στην Python:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
$ sudo rm -rf ~/.cache/pip
```

Βήμα 2ο

Εγκατάσταση των απαραίτητων βιβλιοθηκών για τα εικονικά περιβάλλοντα:

```
$ sudo pip install virtualenv virtualenvwrapper
```

Βήμα 3ο

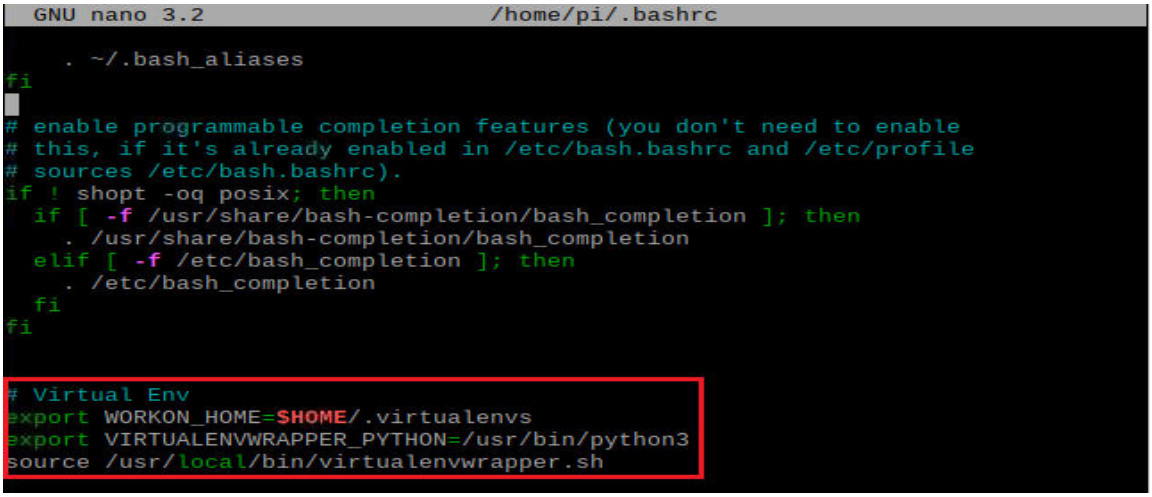
Ανοίγουμε το παρακάτω αρχείο ώστε να προσθέσουμε μερικές εντολές:

```
$ sudo nano ~/.bashrc
```

Βήμα 4ο

Αντιγράφουμε και προσθέτουμε τις παρακάτω εντολές στο τέλος του αρχείου όπως βλέπουμε παρακάτω (Εικόνα 6.6):

```
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```



Εικόνα 6.6 Αρχείο συστήματος Raspberry Pi με όνομα .bashrc

Αποθηκεύουμε το αρχείο πατώντας ταυτόχρονα τα πλήκτρα **'ctrl + x'**, μετά πατάμε το πλήκτρο **'Y'** και τέλος πατάμε **'Enter'**. Τώρα κάθε φορά που θα ανοίγουμε το terminal αυτές οι εντολές θα τρέχουν ενεργοποιώντας τα εικονικά περιβάλλοντα που έχουμε δημιουργήσει έτσι ώστε να είναι έτοιμα για χρήση.

Βήμα 5ο

Εκτελούμε ξανά το αρχείο αφού έχουμε προσθέσει τις παραπάνω εντολές:

```
$ sudo source ~/.bashrc
```

Βήμα 6ο

Δημιουργούμε ένα εικονικό περιβάλλον με όνομα **'virtualexample'** και το εικονικό περιβάλλον μας είναι έτοιμο για χρήση. (Μπορούμε να χρησιμοποιήσουμε ότι όνομα θέλουμε):

```
$ mkvirtualenv virtualexample -p python3
```

Βήμα 7ο

Μπορούμε να απενεργοποιήσουμε το εικονικό μας περιβάλλον δίνοντας την παρακάτω εντολή:

```
$ deactivate
```

Βήμα 8ο

Τέλος αφού και αφού έχουμε ολοκληρώσει με επιτυχία τα προηγούμενα βήματα, μπορούμε να έχουμε πρόσβαση κάθε φορά στο εικονικό μας περιβάλλον μόνο με την παρακάτω εντολή (Όπου **'virtualexample'** το όνομα του εικονικού περιβάλλοντος που δημιουργήσαμε παραπάνω. Σε κάθε περίπτωση μπορούμε να έχουμε πρόσβαση σε οποιοδήποτε άλλο εικονικό περιβάλλον):

```
$ workon virtualexample
```

6.4 Εγκατάσταση απαραίτητων βιβλιοθηκών/πακέτων

6.4.1 Εγκατάσταση βιβλιοθηκών

Παρακάτω θα δημιουργήσουμε το αρχείο requirements.txt, το οποίο θα περιέχει όλες τις απαραίτητες βιβλιοθήκες που θα χρειαστούμε έτσι ώστε να μπορέσουμε να τρέξουμε το πρόγραμμα μας. Έπειτα, θα εκτελέσουμε αυτό το αρχείο έτσι ώστε να πραγματοποιηθούν οι εγκαταστάσεις των βιβλιοθηκών, χωρίς να χρειάζεται να εγκαθιστούμε κάθε βιβλιοθήκη ξεχωριστά.

Βήμα 1ο

Αρχικά αφού βρισκόμαστε στην επιφάνεια εργασίας του Raspberry Pi ανοίγουμε ένα τερματικό (terminal) (Εικόνα 6.7).

Εικόνα 6.7 Τερματικό Raspberry Pi

Βήμα 2ο

Έπειτα εφόσον δεν βρισκόμαστε στο εικονικό μας περιβάλλον δίνουμε την παρακάτω εντολή έτσι ώστε να εισέλθουμε σε αυτό (Προαιρετικά):

```
$ workon virtualexample
```

Βήμα 3ο

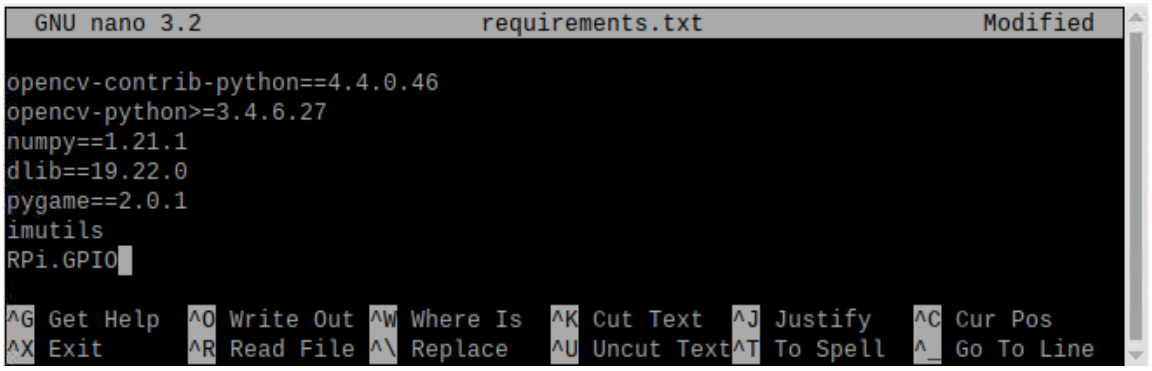
Έπειτα πηγαίνουμε στην επιφάνεια εργασίας και με την εντολή nano δημιουργούμε το αρχείο requirements.txt εκτελώντας τις παρακάτω εντολές:

```
$ cd Desktop
$ nano requirements.txt
```

Βήμα 4ο

Αφού έχουμε ανοίξει το αρχείο (requirements.txt) επιλέγουμε τις παρακάτω εντολές με **'Ctrl + C'** και έπειτα πηγαίνουμε στο τερματικό μας και πατάμε ταυτόχρονα τα πλήκτρα **'Ctrl + Shift + V'** ώστε να κάνουμε επικόλληση (Εικόνα 6.8):

```
opencv-contrib-python==4.4.0.46
opencv-python>=3.4.6.27
numpy==1.21.1
dlib==19.22.0
pygame==2.0.1
picamera[array]
RPi.GPIO
imutils
mediapipe
pyserial
```



Εικόνα 6.8 Αρχείο requirements.txt όπου περιέχει τις βιβλιοθήκες για εγκατάσταση

Αφού έχουμε κάνει επικόλληση αποθηκεύουμε πατώντας ταυτόχρονα τα πλήκτρα **'Ctrl + o'** μετά πατάμε **'Enter'** και επιστρέφουμε στο terminal με τα πλήκτρα **'Ctrl + X'**.

Βήμα 5ο

Για να πραγματοποιήσουμε την εγκατάσταση των βιβλιοθηκών, εκτελούμε την παρακάτω εντολή, και περιμένουμε λίγη ώρα μέχρι να πραγματοποιηθούν όλες οι εγκαταστάσεις:

```
$ pip3 install -r requirements.txt
```

6.4.2 Εγκατάσταση πακέτων

Παρακάτω θα δημιουργήσουμε το αρχείο dependencies.sh το οποίο θα περιέχει όλα τα απαραίτητα πακέτα που θα χρειαστούμε έτσι ώστε να μπορέσουμε να τρέξουμε το πρόγραμμα μας. Έπειτα θα εκτελέσουμε αυτό το αρχείο έτσι ώστε να πραγματοποιηθούν οι εγκαταστάσεις των πακέτων χωρίς να χρειάζεται να εγκαθιστούμε κάθε πακέτο ξεχωριστά.

Βήμα 1ο

Αρχικά υποθέτουμε ότι έχουμε εκτελέσει τα παραπάνω βήματα (2.4.1) και συνεχίζουμε να εκτελούμε διαδοχικά τα υπόλοιπα βήματα στο τερματικό.

Βήμα 2ο

Αφού βρισκόμαστε στο στην επιφάνεια εργασίας δίνουμε την εντολή nano ώστε να δημιουργήσουμε το αρχείο dependencies.sh

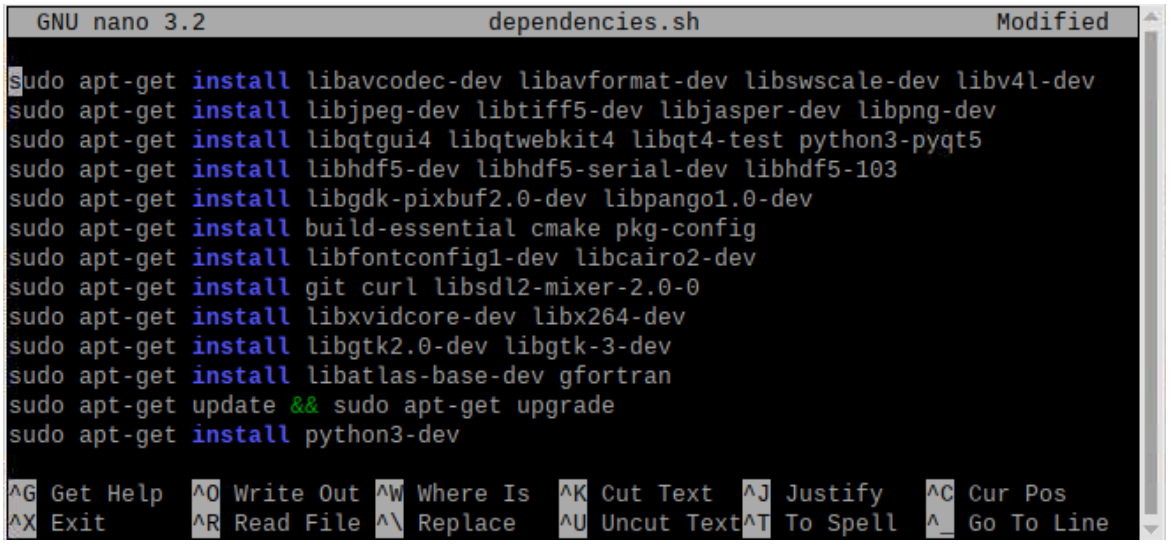
```
$ nano dependencies.sh
```

Βήμα 3ο

Αφού έχουμε ανοίξει το αρχείο (dependencies.sh) επιλέγουμε τις παρακάτω εντολές με **'Ctrl + C'** και έπειτα πηγαίνουμε στο τερματικό μας και πατάμε ταυτόχρονα τα πλήκτρα **'Ctrl + Shift + V'** ώστε να κάνουμε επικόλληση (Εικόνα 6.9):

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libfontconfig1-dev libcairo2-dev
```

```
sudo apt-get install git curl libsdl2-mixer-2.0-0
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev libgtk-3-dev
sudo apt-get install libatlas-base-dev gfortran
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install python3-dev
```



Εικόνα 6.9 Αρχείο dependencies.sh όπου περιέχει τα πακέτα για εγκατάσταση

Αφού έχουμε κάνει επικόλληση αποθηκεύουμε πατώντας ταυτόχρονα τα πλήκτρα **'Ctrl + o'** μετά πατάμε **'Enter'** και επιστρέφουμε στο terminal με τα πλήκτρα **'Ctrl + X'**.

Βήμα 4ο

Δίνουμε όλα τα δικαιώματα στο αρχείο dependencies.sh ώστε να μπορούμε να το εκτελέσουμε:

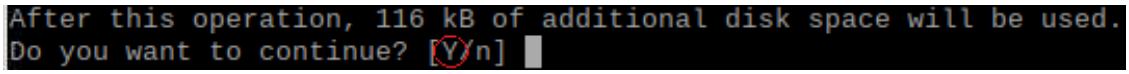
```
$ chmod 777 dependencies.sh
```

Βήμα 5ο

Για να πραγματοποιήσουμε την εγκατάσταση των πακέτων εκτελούμε την παρακάτω εντολή και περιμένουμε λίγη ώρα μέχρι να πραγματοποιηθούν οι εγκαταστάσεις:

```
$ ./dependencies.sh
```

Κατά την εγκατάσταση θα βλέπουμε συχνά το παρακάτω μήνυμα, στο οποίο **θα πρέπει** να δίνουμε το γράμμα **'Y'** ή **'y'** και έπειτα **'Enter'** (Εικόνα 6.10).



Εικόνα 6.10 Μήνυμα κατά την διάρκεια εγκατάστασης των πακέτων

6.5 Σειριακή επικοινωνία του Raspberry Pi με A9G GSM/GPRS+GPS Module

Για να μπορέσουμε να επικοινωνούμε και να δίνουμε εντολές στο A9G Module, θα πρέπει να κάνουμε κάποιες αλλαγές στο σύστημα μας ώστε αυτό να είναι εφικτό. Ο τρόπος που θα επικοινωνεί το Raspberry Pi με το A9G Module θα είναι σειριακά μέσω των UART RX & TX GPIO ακροδεκτών.

Βήμα 1ο

Ανοίγουμε το αρχείο config.txt για επεξεργασία:

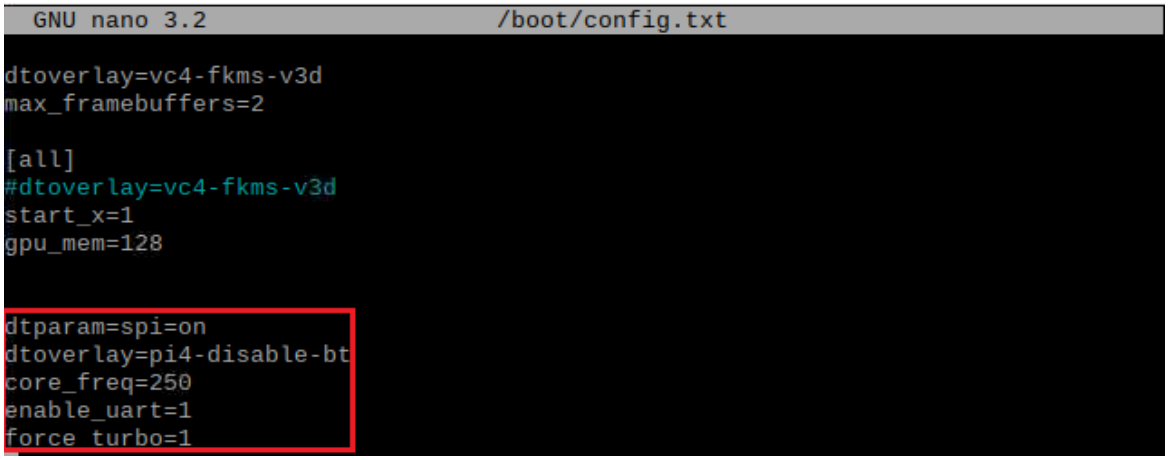
```
$ sudo nano /boot/config.txt
```

Το αρχείο config.txt εκτελείται κάθε φορά που ενεργοποιούμε το σύστημα μας και από εκεί διαβάζει ορισμένες παραμέτρους, τις οποίες θα χρησιμοποιεί κατά την διάρκεια της λειτουργίας του.

Βήμα 2ο

Αφού έχουμε ανοίξει το αρχείο (config.txt) επιλέγουμε τις παρακάτω εντολές με **'Ctrl + C'** και έπειτα πηγαίνουμε στο τέλος του τερματικού μας και πατάμε ταυτόχρονα τα πλήκτρα **'Ctrl + Shift + V'** ώστε να κάνουμε επικόλληση (Εικόνα 6.11):

```
dtparam=spi=on
dtoverlay=pi4-disable-bt
core_freq=250
enable_uart=1
force_turbo=1
```



Εικόνα 6.11 Αρχείο config.txt με επιπλέον ρυθμίσεις

Αφού έχουμε κάνει επικόλληση αποθηκεύουμε πατώντας ταυτόχρονα τα πλήκτρα **'Ctrl + o'** μετά πατάμε **'Enter'** και επιστρέφουμε στο terminal με τα πλήκτρα **'Ctrl + X'**.

Βήμα 3ο

Δημιουργούμε ένα αντίγραφο ασφαλείας (cmdline_backup.txt) του αρχείο cmdline.txt:

```
$ sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt
```

Ανοίγουμε το αρχείο cmdline.txt για επεξεργασία:

```
$ sudo nano /boot/cmdline.txt
```

Διαγράφουμε ότι υπάρχει μέσα στο αρχείο (cmdline.txt), επιλέγουμε την παρακάτω εντολή με **'Ctrl + C'** και έπειτα πηγαίνουμε στο τερματικό μας και πατάμε ταυτόχρονα τα πλήκτρα **'Ctrl + Shift + V'** ώστε να κάνουμε επικόλληση.

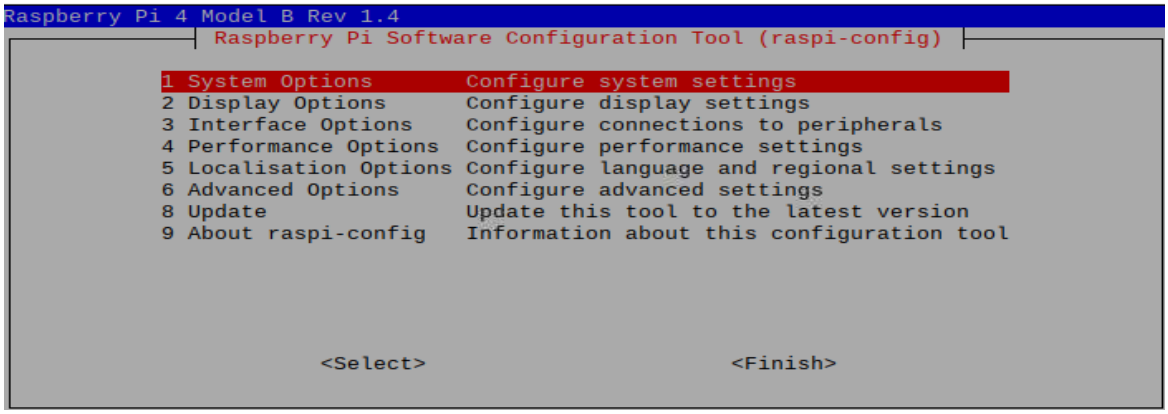
```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
```

Η εντολή θα πρέπει να βρίσκεται σε μια γραμμή στο αρχείο cmdline.txt. Αφού έχουμε κάνει επικόλληση αποθηκεύουμε πατώντας ταυτόχρονα τα πλήκτρα **'Ctrl + o'** μετά πατάμε **'Enter'** και επιστρέφουμε στο terminal με τα πλήκτρα **'Ctrl + X'**.

Βήμα 4ο

Δίνουμε την παρακάτω εντολή ώστε να ανοίξουμε τις ρυθμίσεις του συστήματος (Εικόνα 6.12):

```
$ sudo raspi-config
```



Εικόνα 6.12 Ρυθμίσεις Συστήματος

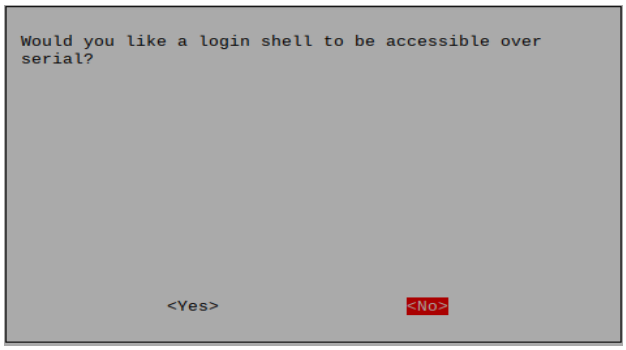
Πηγαίνουμε στην επιλογή '**3 Interface Options'** και πατάμε **'Enter'**(Εικόνα 6.13)**:**

Εικόνα 6.13 Επιλογή διαχείρισης διεπαφών

Πηγαίνουμε στην επιλογή '**P6 Serial Port'** και πατάμε **'Enter'**(Εικόνα 6.14)**:**

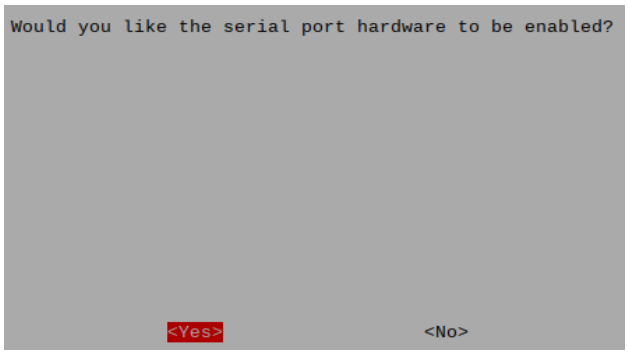
Εικόνα 6.14 Επιλογή διαχείρισης σειριακής πόρτας

Επιλέγουμε **'No'** και πατάμε **'Enter'**(Εικόνα 6.15):



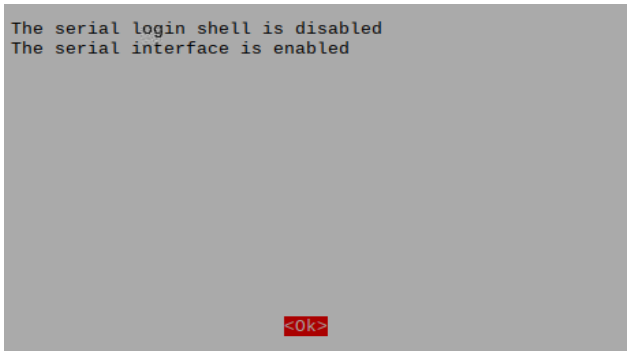
Εικόνα 6.15 Ερώτηση ενεργοποίησης login shell

Επιλέγουμε **'Yes'** και πατάμε **'Enter'**(Εικόνα 6.16):



Εικόνα 6.16 Ερώτηση ενεργοποίησης serial port hardware

Τέλος πατάμε **'Enter'** και κλείνουμε το τερματικό(Εικόνα 6.17):



Εικόνα 6.17 Εμφάνιση επιλογών ρύθμισης

Βήμα 5ο

Τέλος κάνουμε μια επανεκκίνηση του συστήματος μας ώστε να εφαρμοστούν όλες οι παραπάνω ρυθμίσεις που πραγματοποιήσαμε:

```
$ sudo reboot
```

Βήμα 6ο

6.1 Εγκαθιστούμε το picocom εφόσον δεν υπάρχει:

```
$ sudo apt-get install picocom
```

Το picocom είναι ένα πρόγραμμα εξομοίωσης, το οποίο μας δίνει την δυνατότητα για τον συγχρονισμό και την επικοινωνία μέσω ενός τερματικού, με μια σειριακή θύρα ενός Linux συστήματος.

6.2 Μόλις το σύστημα μας είναι έτοιμο ανοίγουμε ένα τερματικό και εκτελούμε την παρακάτω εντολή:

```
$ sudo picocom --baud 9600 /dev/ttyS0
```

Η εντολή δίνει την δυνατότητα επικοινωνίας με την σειριακή θύρα ttyS0 με ρυθμό 9600 baud.

6.3 Εφόσον έχουν συνδεθεί ίσως χρειαστεί να περιμένουμε λίγο μέχρι να μπορέσουμε να εκτελέσουμε AT εντολές. Δίνουμε 2-3 φορές την εντολή **'AT'** και περιμένουμε να λάβουμε ως απάντηση **'OK'**:

```
AT
# OK
```

Η εντολή **'AT'** χρησιμοποιείται για να δούμε αν υπάρχει επικοινωνία του συστήματος μας με το A9G Module. Το αποτέλεσμα που θα πρέπει να λάβουμε για να συνεχίσουμε θα πρέπει να είναι **'OK'** σε κάθε άλλη περίπτωση θα πρέπει να ξανά ακολουθήσουμε να παραπάνω βήματα και να ελέγξουμε για τυχών λάθη στην συνδεσμολογία του κυκλώματος μας.

6.6 Πρόσβαση του Raspberry Pi στο διαδίκτυο μέσω του A9G GSM/GPRS+GPS Module

Παρακάτω θα δημιουργήσουμε και θα τροποποιήσουμε τα απαραίτητα αρχεία, έτσι ώστε να μπορέσουμε να επιτύχουμε την σύνδεση του Raspberry Pi στο διαδίκτυο χρησιμοποιώντας το A9G Module. Σκοπός, είναι να έχουμε πρόσβαση στο διαδίκτυο, ώστε να μπορούμε να κάνουμε χρήση δεδομένων όπου αυτό είναι εφικτό, μέσω του δικτύου κινητής τηλεφωνίας.

Βήμα 1ο

Αρχικά η κάρτα SIM που έχουμε τοποθετήσει στο A9G Module είναι στην περίπτωση μας της **Vodafone** και σε αυτήν την περίπτωση το **όνομα σημείου πρόσβασης (APN)** για την πρόσβαση στο internet είναι το '**internet.vodafone.gr'**. Σε κάθε περίπτωση θα πρέπει να γνωρίζουμε το όνομα σημείου πρόσβασης (APN) της εκάστοτε εταιρίας τηλεπικοινωνιών που χρησιμοποιούμε (Πίνακας 4).

Εταιρεία Τηλεπικοινωνιών	**APN**
VODAFONE	internet.vodafone.gr
COSMOTE	internet
WIND	gint.b-online.gr

Πίνακας 4 Αντιστοίχηση εταιριών τηλεπικοινωνιών με τα APN τους

Βήμα 2ο

Κάνουμε ένα update το σύστημα μας, και ύστερα εγκαθιστούμε το λογισμικό ppp:

```
$ sudo apt-get update
$ sudo apt-get install ppp screen elinks
```

Βήμα 3ο

3.1 Με την παρακάτω εντολή συνδεόμαστε ως root ώστε να μπορέσουμε να τροποποιήσουμε κάποια αρχεία και να δώσουμε τις απαραίτητες εντολές:

```
$ sudo -i
```

3.2 Πηγαίνουμε στο κατάλογο peers:

```
$ cd /etc/ppp/peers/
```

3.3 Δημιουργούμε το αρχείο **rnet**:

```
$ nano rnet
```

3.4 Αφού έχουμε ανοίξει το αρχείο (rnet) επιλέγουμε τις παρακάτω εντολές με **'Ctrl + C'** και έπειτα πηγαίνουμε στο τερματικό μας και πατάμε ταυτόχρονα τα πλήκτρα **'Ctrl + Shift + V'** ώστε να κάνουμε επικόλληση:

```
# My APN internet.vodafone.gr
connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs -T internet.vodafone.gr"

# Communication port
/dev/ttyS0

# Baudrate
9600

# Assumes that your IP address is allocated dynamically by the ISP.
noipdefault

# Try to get the name server addresses from the ISP.
usepeerdns

# Use this connection as the default route to the internet.
defaultroute

# Makes PPPD "dial again" when the connection is lost.
persist

# Do not ask the remote to authenticate.
noauth

# No hardware flow control on the serial link with GSM Modem
nocrtscts
# No modem control lines with GSM Modem
local
```

Βλέπουμε πως στην δεύτερη γραμμή όπου πραγματοποιείται το **connect** έχουμε εισάγει το **APN της Vodafone**, οπότε **σε κάθε περίπτωση** εισάγουμε και το **αντίστοιχο APN** που χρησιμοποιούμε. Αφού έχουμε κάνει επικόλληση, και τις απαραίτητες τροποποιήσεις, αποθηκεύουμε πατώντας ταυτόχρονα τα πλήκτρα **'Ctrl + o'** μετά πατάμε **'Enter'** και επιστρέφουμε στο terminal με τα πλήκτρα **'Ctrl + X'**.

3.5 Ακόμα, θα πρέπει να επεξεργαστούμε το αρχείο gsm.py και να εισάγουμε το δικό μας αντίστοιχο APN, όπως βλέπουμε στην ενότητα 7.4 Επεξήγηση βασικών τμημάτων κώδικα κάθε αρχείου → gsm.py → Γραμμή 57.

Βήμα 4° (Πραγματοποιούμε το βήμα αυτό εφόσον η κάρτα SIM που χρησιμοποιούμε έχει 4ψήφιο κωδικό, σε άλλη περίπτωση παραλείπουμε το βήμα αυτό)

4.1 Πηγαίνουμε στο κατάλογο chatscripts:

```
$ cd /etc/chatscripts/
```

4.2 Ανοίγουμε το αρχείο **gprs** για επεξεργασία:

```
$ nano gprs
```

4.3 Εφόσον η κάρτα SIM που χρησιμοποιούμε έχει κωδικό τον εισάγουμε σε αυτό το αρχείο όπως φαίνεται στην Εικόνα 6.18.

```
# cease if the modem is not attached to the network yet
ABORT           "+CGATT: 0"

""              AT
TIMEOUT         12
OK              ATH
OK              ATE1

# +CPIN provides the SIM card PIN
OK              AT+CPIN=2255

# +CFUN may allow to configure the handset to limit operations to
# GPRS/EDGE/UMTS/etc to save power, but the arguments are not standard
# except for 1 which means "full functionality".
#OK             AT+CFUN=1

OK              AT+CGDCONT=1,"IP","\T","",0,0
OK              ATD*99#
TIMEOUT         22
CONNECT         ""
```

Εικόνα 6.18 Εισαγωγή PIN της κάρτας SIM

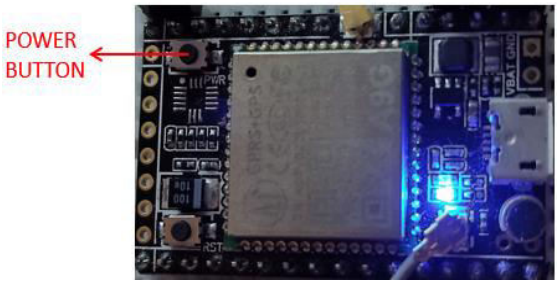
Αφού έχουμε εισάγει τον κωδικό αποθηκεύουμε πατώντας ταυτόχρονα τα πλήκτρα **'Ctrl + o'** μετά πατάμε **'Enter'** και επιστρέφουμε στο terminal με τα πλήκτρα **'Ctrl + X'**.

4.4 Ακόμα, θα πρέπει να επεξεργαστούμε το αρχείο gsm.py και να εισάγουμε το δικό μας κωδικό, όπως βλέπουμε στην ενότητα 7.4 Επεξήγηση βασικών τμημάτων κώδικα κάθε αρχείου → gsm.py → Γραμμή 42-43.

Βήμα 5^{ο} (Προαιρετικό βήμα. Καθώς ενεργοποιούμε αυτή την λειτουργία μέσω του προγράμματος μας)

Πραγματοποιούμε εφόσον είτε θέλουμε να ενεργοποιήσουμε το A9G Module χειροκίνητα ώστε να δοκιμάσουμε την ορθή λειτουργία του GSM προτού τρέξουμε το πρόγραμμα μας, είτε να χρησιμοποιήσουμε το GSM ανεξάρτητα από το πρόγραμμα, ώστε να συνδεθούμε και να χρησιμοποιήσουμε υπηρεσίες διαδικτύου.

5.1 Προτού εκκινήσουμε το αρχείο, βεβαιωνόμαστε πως η μπλε λυχνία του A9G Module μας είναι σταθερά ανοιχτή (Εικόνα 6.19). Σε κάθε άλλη περίπτωση ενεργοποιούμε το A9G Module μας.



Εικόνα 6.19 A9G Module σε ενεργοποιημένη κατάσταση

5.2 Πηγαίνουμε στο κατάλογο peers:

```
$ cd /etc/ppp/peers/
```

5.3 Εκκινούμε το αρχείο **rnet** που δημιουργήσαμε:

```
$ pon rnet
```

5.4 Περιμένουμε λίγο και εκτελούμε την παρακάτω εντολή έτσι ώστε να δούμε εάν η εκτέλεση του αρχείου έχει πραγματοποιηθεί επιτυχώς:

```
$ cat /var/log/syslog | grep pppd
```

Το αναμενόμενο αποτέλεσμα που περιμένουμε να λάβουμε για την ορθή εκκίνηση και λειτουργία είναι αυτό που βλέπουμε στην Εικόνα 6.20.

```
raspberrypi pppd[804]: pppd 2.4.7 started by pi, uid 0
raspberrypi pppd[804]: Serial connection established.
raspberrypi pppd[804]: Using interface ppp0
raspberrypi pppd[804]: Connect: ppp0 <--> /dev/ttyS0
raspberrypi pppd[804]: Remote message: Login OK
raspberrypi pppd[804]: PAP authentication succeeded
raspberrypi pppd[804]: local  IP address [redacted]
raspberrypi pppd[804]: remote IP address [redacted]
raspberrypi pppd[804]: primary   DNS address [redacted]
raspberrypi pppd[804]: secondary DNS address [redacted]
```

Εικόνα 6.20 Επιτυχής εγκαθίδρυση σύνδεσης

Σε περίπτωση που βλέπουμε το παρακάτω μήνυμα αποτυχίας (Εικόνα 6.21), τότε:

```
pi@raspberrypi:/etc/ppp/peers $ cat /var/log/syslog | grep pppd
Oct 13 00:01:54 raspberrypi pppd[1066]: pppd 2.4.7 started by pi, uid 0
Oct 13 00:02:07 raspberrypi pppd[1066]: Connect script failed
Oct 13 00:02:49 raspberrypi pppd[1066]: Connect script failed
Oct 13 00:03:31 raspberrypi pppd[1066]: Connect script failed
Oct 13 00:04:13 raspberrypi pppd[1066]: Connect script failed
Oct 13 00:04:56 raspberrypi pppd[1066]: Connect script failed
Oct 13 00:05:38 raspberrypi pppd[1066]: Connect script failed
Oct 13 00:06:20 raspberrypi pppd[1066]: Connect script failed
```

Εικόνα 6.21 Αποτυχία εκτέλεσης αρχείου

5.4.1 Θα πρέπει να ελέγξουμε εάν το A9G Module βρίσκεται ενεργοποιημένο όπως είδαμε (Εικόνα 6.19)

5.4.2 Θα πρέπει να ελέγξουμε εάν το Raspberry Pi εμφανίζει κάποιο είδους προειδοποιητικό μήνυμα όπως βλέπουμε στην Εικόνα 6.22, το οποίο αφορά την προειδοποίηση της χαμηλής τάσης του Raspberry Pi, το οποίο επηρεάζει άμεσα το A9G Module, καθώς δεν τροφοδοτείται επαρκώς ώστε να λειτουργεί σωστά.

Εικόνα 6.22 Προειδοποίηση χαμηλής τάσης του Raspberry Pi

5.4.3 Αλλιώς επαναλάβουμε τα παραπάνω βήματα (από Βήμα 1^{ο}) σε περίπτωση που έχει προηγηθεί κάποιο λάθος.

5.4.4 Στην περίπτωση που κάτι από τα παραπάνω δεν λειτουργεί, και έχουμε κινητή συσκευή που υποστηρίζει Hotspot λειτουργία, είναι προτιμότερο να ανοίξουμε τα δεδομένα στο κινητό, να ενεργοποιήσουμε την λειτουργία Hotspot, έπειτα να ενεργοποιήσουμε το Wi-Fi στο Raspberry Pi, και τέλος να συνδεθούμε στην συσκευή μας, ώστε να έχουμε πρόσβαση στο διαδίκτυο.

6.7 Τροποποίηση στοιχείων στο αρχείο mail_credentials.txt

Το αρχείο **'mail_credentials.txt'** περιέχει στοιχεία τα οποία θα πρέπει να συμπληρώσουμε προτού εκκινήσουμε το πρόγραμμα μας.

Βλέπουμε παρακάτω τα περιεχόμενα του αρχείου:

```
Driver's Name  : <Write the driver's name here>
My email       : <Write the sender's email here(gmail)>
My password    : <Write the sender's password for the email here>
Receiver email : <Write the receiver's email here(whatever mail server)>
```

Driver's Name: Το όνομα του οδηγού

My email: Το email αποστολέα από το οποίο θέλουμε να στείλουμε το μήνυμα. **ΠΡΟΣΟΧΗ** το email θα πρέπει να έχει κατάληξη **'gmail.com'**, καθώς στον κώδικα μας έχουμε ορίσει smtp.gmail.com server.

Driver's Name: Ο κωδικός του email αποστολέα.

Driver's Name: Το email του παραλήπτη το οποίο μπορεί να ανήκει σε διαφορετικό πάροχο email (π.χ. Outlook, Yahoo κτλ.).

Παράδειγμα συμπλήρωσης περιεχόμενων αρχείου:

```
Driver's Name  : Christos
My email       : christosmail@gmail.com
My password    : 1234
Receiver email : christosmail@hotmail.com
```

7. Ανάλυση του Έργου

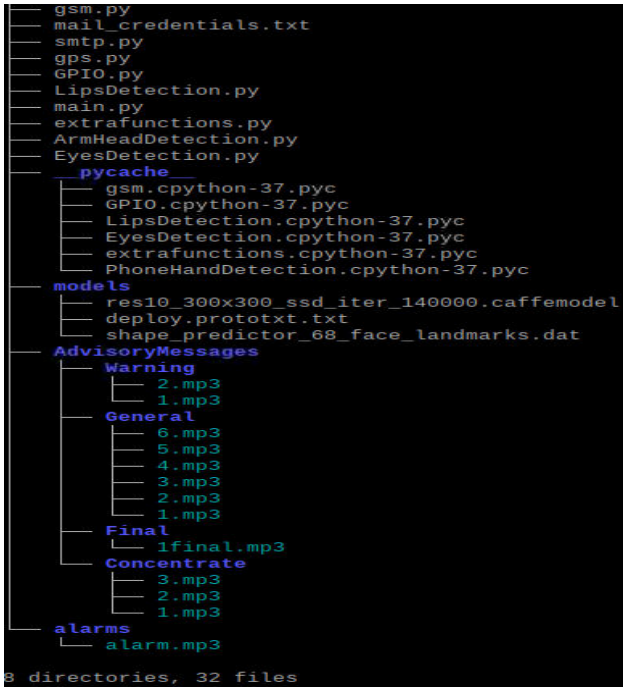
7.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα αναλύσουμε τις λειτουργίες του έργου μας, και θα δούμε τη σκοπό είχαν αυτές, ώστε το πρόγραμμα μας να είναι σε θέση να μπορεί να παρακολουθεί και να εντοπίζει υπνηλία, ανήσυχη/περίεργη συμπεριφορά και νευρικότητα όπως αρχικά έχουμε αναφέρει. Έπειτα θα δούμε πως μέσω e-mail, μπορούμε να στέλνουμε την τοποθεσία και την κατάσταση του οδηγού. Θα κοιτάξουμε τις πρακτικές διαφορές στα μοντέλα Haar cascades και Caffe model, καθώς έχουμε ήδη μελετήσει τα μοντέλα θεωρητικά στο κεφάλαιο 3.2 Τεχνικές Ανίχνευσης Προσώπων. Τέλος θα θέσουμε το σύστημα μας σε λειτουργία, και θα δούμε πραγματικά αποτελέσματα, κατά την διάρκεια της χρήσης του.

7.2 Δομή αρχείων έργου

Παρακάτω θα δούμε την δομή των αρχείων η οποία συμπεριλαμβάνει όλα τα απαραίτητα αρχεία για την εκτέλεση του έργου μας.

Ενώ βρισκόμαστε στον κατάλογο όπου βρίσκεται το έργο μας, δίνουμε την εντολή **$ tree -c -r** έτσι ώστε να δούμε την διαδρομή όλων των αρχείων σε μορφή δένδρου (Εικόνα 7.1).



Εικόνα 7.1 Προβολή αρχείων του έργου μας μέσω της εντολή tree

7.2.1 Περιγραφή δομής αρχείων και καταλόγων

main.py – Το κυρίως εκτελέσιμο πρόγραμμα, το οποίο εισάγει όλες τις απαραίτητες κλάσεις και βιβλιοθήκες, καθώς και όλη την βάση του προγράμματος.

EyesDetection.py – Κλάση η οποία περιέχει αρχικοποιήσεις και μεθόδους, οι οποίες δρουν ανάλογα, στην περίπτωση που οδηγός έχει κλειστά τα μάτια του.

LipsDetection.py – Κλάση η οποία περιέχει αρχικοποιήσεις και μεθόδους, οι οποίες λειτουργούν ανάλογα, στην περίπτωση όπου ο οδηγός χασμουριέται.

ArmHeadDetection.py – Κλάση η οποία περιέχει αρχικοποιήσεις και μεθόδους, οι οποίες χρησιμοποιούνται για να εντοπίσουν περίεργες συμπεριφορές του οδηγού. Τέτοιες συμπεριφορές παραδείγματος χάρη είναι ο οδηγός να μιλά στο κινητό, να έχει το ένα ή και τα δύο του χέρια σε μη κανονική θέση (θεωρούμε κανονική θέση, τα δύο χέρια του οδηγού να βρίσκονται στο τιμόνι), και επίσης το να βλέπουμε το κεφάλι του να έχει μια κλίση ως προς την αριστερή ή δεξιά πλευρά, δηλαδή να πλησιάζει τον ώμο του(που σημαίνει ότι ο οδηγός είναι είτε κουρασμένος είτε εμφανίζει σημάδια περίεργης συμπεριφοράς).

extrafunctions.py – Αρχείο το οποίο περιέχει βασικές συναρτήσεις όπως αλλαγή μεγέθους πλαισίου και συναρτήσεις που υπολογίζουν την απόσταση του οδηγού από την κάμερα.

smtp.py – Αρχείο το οποίο περιέχει συναρτήσεις οι οποίες μας δίνουν την δυνατότητα να αποστείλουμε ηλεκτρονικά μηνύματα, σχετικά με την τρέχουσα τοποθεσία του οδηγού.

gps.py – Αρχείο το οποίο περιέχει 2 συναρτήσεις, η μια συνάρτηση η οποία αφορά την ενεργοποίηση της GPS λειτουργίας στο A9G Module, και η άλλη συνάρτηση η οποία αφορά την ανάκτηση γεωγραφικών δεδομένων, όπως το γεωγραφικό πλάτος και μήκος.

gsm.py – Αρχείο το οποίο περιέχει συνάρτηση η οποία αφορά την ενεργοποίηση της GSM λειτουργίας του A9G Module και την σύνδεση του στο διαδίκτυο, με σκοπό την χρήση των δεδομένων κινητής τηλεφωνίας.

GPIO.py – Αρχείο το οποίο περιέχει συνάρτησης αρχικοποίησης για τις διόδους φωτοεκπομπής (LED), συναρτήσεις για την ενεργοποίηση κάθε χρώματος (Πράσινο,

Κόκκινο, Κίτρινο) και συναρτήσεις για την ενεργοποίηση και την επαναφορά του A9G Module.

models – Κατάλογος ο οποίος περιέχει το αρχείο .protxt (deploy.prototxt.txt), το οποίο ορίζει την αρχιτεκτονική του μοντέλου, μαζί περιέχει το αρχείο .caffemodel (res10_300x300_ssd_iter_140000.caffemodel), το οποίο αφορά τα βάρη που θα χρησιμοποιήσουν τα επίπεδα (layers) του μοντέλου. Τέλος περιέχει το αρχείο (shape_predictor_68_face_landmarks.dat), που χρησιμοποιείται για τον εντοπισμό 68 σημείων του προσώπου, όπου χωρίζονται σε 5 κατηγορίες (Μάτια, Φρύδια, Μύτη, Χείλη/Στόμα και Πιγούνι).

mail_credentials.txt – Αυτό το αρχείο θα πρέπει να περιέχει το όνομα του οδηγού, τα απαραίτητα διαπιστευτήρια σύνδεσης του email αποστολέα και το email αποστολής του παραλήπτη.

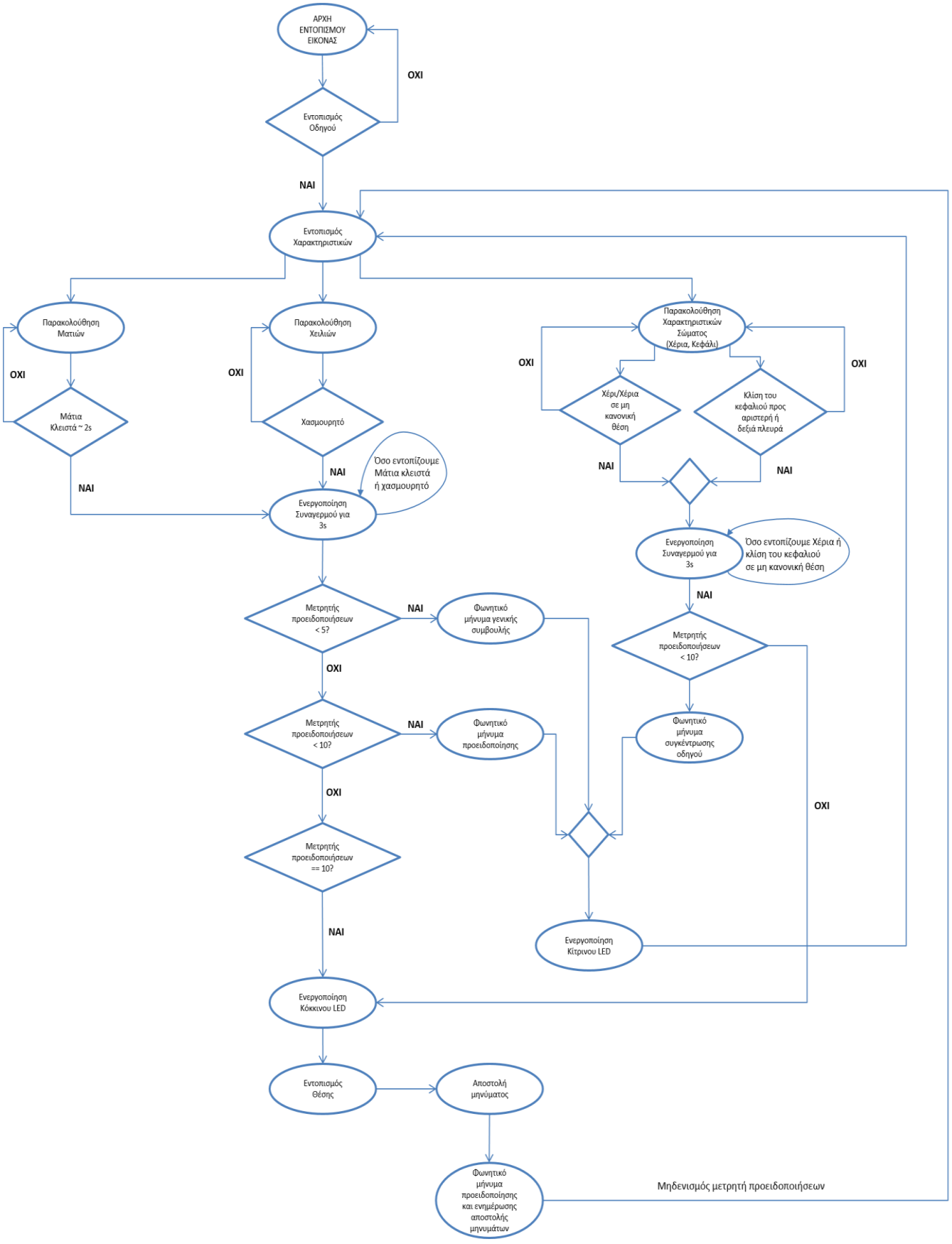
AdvisoryMessages – Κατάλογος ο οποίος περιέχει 4 υποκαταλόγους οι οποίοι περιέχουν φωνητικά μηνύματα συμβουλευτικής φύσεως.

- **Concentrate** - Υποκατάλογος που περιέχει 3 αρχεία (.mp3) φωνητικών εντολών, τα οποία δίνουν συμβουλές στον οδηγό, έτσι ώστε να προσέχει και να παραμένει πιο συγκεντρωμένος.
- **General** - Υποκατάλογος που περιέχει 6 αρχεία (.mp3) φωνητικών εντολών, τα οποία δίνουν γενικές συμβουλές στον οδηγό, του τύπου να κάνει διαλείμματα, να ξεκουράζεται κτλ.
- **Warning** – Υποκατάλογος που περιέχει 2 αρχεία (.mp3) φωνητικών εντολών, τα οποία χρησιμοποιούνται για να προειδοποιούν τον οδηγό, πως φαίνεται πολύ κουρασμένος και θα πρέπει να σταματάει να ξεκουράζεται.
- **Final** - Υποκατάλογος που περιέχει 1 αρχείο (.mp3) φωνητικής εντολής, το οποίο αναπαράγεται μόλις ο οδηγός έχει ξεπεράσει ένα συγκεκριμένο όριο προειδοποιήσεων, τότε ο οδηγός προειδοποιείτε για την κατάσταση του και ενημερώνεται πως έχει σταλθεί μήνυμα στα συγγενικό του πρόσωπο.

__pycache__ – Κατάλογος ο οποίος δημιουργείται κατά την διαδικασία εκτέλεσης των προγραμμάτων μας. Όπως είδαμε και στην ενότητα 4.3, όταν θέλουμε να εκτελέσουμε ένα πρόγραμμα, ο διερμηνευτής την python μεταγλωττίζει πρώτα το αρχείο σε μορφή bytes (.pyc ή .pyo) και ύστερα το αποθηκεύει στο κατάλογο __pycache__. Αυτός ο

κατάλογος δίνει την δυνατότητα στο πρόγραμμα μας να ξεκινήσει λίγο πιο γρήγορα κάθε φορά, αλλά σε περίπτωση που αλλάξουμε κάτι σε κάποιο από τα αρχεία μας (.py) θα πρέπει τα αρχεία αυτά να δημιουργηθούν εκ νέου. Επομένως είτε διαγράψουμε αυτό τον κατάλογο είτε το αφήσουμε ως έχει, από την στιγμή που θα τρέξουμε το πρόγραμμα μας ο φάκελος __pycache__ είτε θα δημιουργηθεί από την αρχή περιέχοντας τα απαραίτητα bytes αρχεία, είτε κατά την μεταγλωττίσει τα bytes αρχεία θα τροποποιηθούν αναλόγως.

7.3 UML Διάγραμμα δραστηριότητας του έργου μας



7.4 Επεξήγηση βασικών τμημάτων κώδικα κάθε αρχείου

➢ **EyesDetection.py**

○ Συνάρτηση **eye_aspect_ratio**:

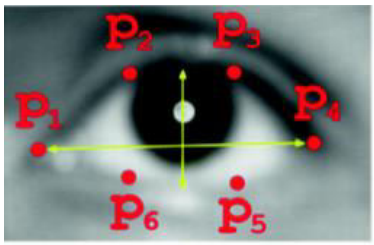
```
"""
Calculates the aspect ratio from 6 points of eyes
and return it
"""
def eye_aspect_ratio(self, eye):

      A = np.linalg.norm(eye[1] - eye[5])
      B = np.linalg.norm(eye[2] - eye[4])
      C = np.linalg.norm(eye[0] - eye[3])

      ear = (A + B) / (2.0 * C)
      return ear
```

Παράμετροι:

eye – 6 σημεία(x,y) τα οποία βρίσκονται γύρω από το μάτι (Εικόνα 7.2 (α))

Επεξήγηση συνάρτησης:

Η συνάρτηση όπως βλέπουμε αρχικά, βρίσκει τη διαφορά της απόστασης, μεταξύ των σημείων του ματιού που έχουμε προσδιορίσει. Ύστερα, χρησιμοποιώντας τον τύπο EAR (Eye Aspect Ratio), που βλέπουμε στην Εικόνα 7.2 (β), υπολογίζει και επιστρέφει την διαφορά της απόστασης των σημείων του ματιού, δηλαδή όσο πιο ανοιχτό είναι το μάτι τόσο μεγαλύτερη θα είναι η τιμή, ενώ όσο πιο κλειστό θα είναι το μάτι τόσο μικρότερη θα είναι η τιμή καθώς η δύο κάθετες αποστάσεις (P2-P6 & P3-P5) που υπολογίζουμε τείνουν να έρχονται πιο κοντά μεταξύ τους πλησιάζοντας το οριζόντιο σημείο (P1-P4). Τέλος επιστρέφουμε το αποτέλεσμα.



(α)

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

(β)

Εικόνα 7.2 (α) 6 σημεία του ματιού, (β) Τύπος για τον υπολογισμό της απόστασης των 6 σημείων του ματιού

○ Συνάρτηση **detect_eyes_drowsines**:

```
1 def detect_eyes_drowsiness(self, frame, AvgEar,
  EYE_THRESHOLD, MAX_FRAMES, alarm, TotalDetections, FINAL_FLAG):
```

```
2
3       '''Detect eyes drowsiness
4
5       Args:
6           frame       : Frame to process
7           AvgEar      : Average of Eye Aspect Ratio(E.A.R) for both eyes
8           EYE_THRESHOLD    : The max threshold E.A.R
9           MAX_FRAMES       : Max Frames for eyes
10          alarm            : Alarm sound
11          TotalDetections  : Counter for total time detections
12          FINAL_FLAG       : True when EYES Drowsiness detected
13      '''
```

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία.

AvgEar: Μέση τιμή των ματιών που προκύπτει από την συνάρτηση '**eye_aspect_ratio'** (υπολογίζεται για κάθε μάτι ξεχωριστά και ύστερα θέτουμε την μέση τιμή των ματιών).

EYE_THRESHOLD: Η τιμή κατωφλίου των ματιών.

MAX_FRAMES: Μέγιστος αριθμός πλαισίων τον οποίον αν ξεπεράσουμε εντοπίζουμε κλειστά μάτια.

alarm: Ο συναγερμός που θέτουμε.

TotalDetections: Συνολικός μετρητής εντοπισμών.

FINAL_FLAG: Flag το οποίο γίνεται αληθές, μόλις εντοπίσουμε κλειστά μάτια.

```
15      # If Eye Aspect Ratio(E.A.R) is less than Eye threshold
16      # then increase the counter
17      if AvgEar < EYE_THRESHOLD:
18          self.EYE_COUNTER += 1
19          self.FLAG = True
20          print("EYES COUNTER FRAMES: ", self.EYE_COUNTER)
21
22          # If the eyes remain closed for too many number of frames
23          # then we have detected a drowsiness
24          if self.EYE_COUNTER >= MAX_FRAMES:
25              self.FLAG_LED = True
26              cv2.putText(frame,
                "DROWSINESS ALERT![EYES CLOSED]", (10, 30),
27              cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

Γραμμή 17-19: Αρχικά κοιτάμε αν η μέση τιμή των ματιών, είναι μικρότερη από την τιμή κατωφλίου που έχουμε θέσει. Έπειτα αυξάνουμε τον μετρητή της συνθήκης και κάνουμε αληθές ένα flag, το οποίο δηλώνει ότι βρισκόμαστε στην συνθήκη.

Γραμμή 24-27: Εάν ο μετρητής που αυξήσαμε στην προηγούμενη συνθήκη, ξεπεράσει το μέγιστο αριθμό πλαισίων που έχουμε θέσει, τότε θέτουμε το flag που αφορά το LED σε αληθές, και εμφανίζουμε ένα μήνυμα πως εντοπίσαμε κλειστά μάτια.

```
30              if FINAL_FLAG == False:
31                  # If the counter has crossed the limit,
                 # then we play a warning message
32                  # and at the same time we send a message
                    # to the relatives
33                  if (TotalDetections % 10) == 0
                    and TotalDetections!=0 and self.FSoundFlag == True:
34                      self.FINAL_FLAG = True
35                      self.once_warning += 1
36                      self.FSoundFlag = False
37                      if self.once_warning == 1:
38                          if pygame.mixer.Channel(1).get_busy() == False:
39                              if self.t3.is_alive() is False:
40                                  self.t3=threading.Thread
                                    (target=Red_LED)
                                    self.t3.start()
41                              if self.send_mail.is_alive() is False:
                                    thread_running = False
                                    for thread in threading.enumerate():
                                        if thread.name in self.sendmaillist:
                                            thread_running = True
42                                  if thread_running == False:
                                        self.send_mail =
                                        threading.Thread(target=send_mail)
                                        self.send_mail.name = "SendMailEyes"
                                        self.send_mail.start()

43                              self.FSoundFlag = False
44                              pygame.mixer.stop()
45                              advisory = pygame.mixer.Sound
                            ("AdvisoryMessages/Final/1final.mp3")
46                              advisory.play()
47                      elif self.once_warning == 2:
48                          self.FSoundFlag = False
```

Γραμμή 30: Εάν το flag είναι ψευδές, δηλαδή δεν εντοπίζουμε ήδη κλειστά μάτια συνεχίζουμε.

Γραμμή 33: Εάν το υπόλοιπο του μετρητή (σύνολο εντοπισμών) είναι ίσο με 10 ΚΑΙ ο μετρητής είναι διαφορετικό του 0 ΚΑΙ το flag είναι αληθές, τότε εκτελούμε τις εντολές. Αυτή η συνθήκη χρησιμοποιείται έτσι ώστε να εκτελούνται οι εντολές που περιέχει, μόνο όταν το υπόλοιπο του μετρητή συνόλου εντοπισμών (TotalDetections) είναι ακριβώς ίσο με 0 (π.χ. 10, 20, 30, 40 κτλ.).

Γραμμή 34-36: Θέτουμε 2 flag με αρχική τιμή αληθής, και αυξάνουμε ένα μετρητή.

Γραμμή 37: Βεβαιώνουμε ότι ο μετρητής έχει αυξηθεί μια φορά. Αυτό συμβαίνει για να αποτρέψουμε το γεγονός να εισέλθουμε $2^{η}$ φορά στο ίδιο σημείο του κώδικα.

Γραμμή 38: Βλέπουμε εάν το κανάλι 1 το οποίο αναπαράγει τα φωνητικά μηνύματα, μας επιστρέφει κατάσταση ψευδής, δηλαδή δεν αναπαράγει κάποιο φωνητικό μήνυμα την συγκεκριμένη στιγμή και επομένως μπορούμε να συνεχίσουμε.

Γραμμή 39: Βλέπουμε εάν το νήμα, βρίσκεται σε κατάσταση ψεύδους.

Γραμμή 40: Ενεργοποιούμε το κόκκινο LED.

Γραμμή 41: Αρχικά, κοιτάζουμε εάν το νήμα του όπου στέλνουμε το μήνυμα βρίσκεται σε κατάσταση ψεύδους (δηλαδή δεν τρέχει). Έπειτα τρέχουμε ένα βρόχο ο οποίος διατρέχει όλα τα διαθέσιμα νήματα που τρέχουν, και με αυτόν τον τρόπο ελέγχουμε εάν το νήμα που στέλνει το ηλεκτρονικό μήνυμα, χρησιμοποιείται από κάποια άλλη κλάση εκείνη τη στιγμή. Στην περίπτωση που αυτό το νήμα τρέχει από κάποια άλλη κλάση θέτουμε την μεταβλητή thread_running ως αληθής.

Γραμμή 42: Εάν η μεταβλητή thread_running είναι ψευδής (δηλαδή το νήμα που αφορά την αποστολή του ηλεκτρονικού μηνύματος δεν τρέχει από άλλη κλάση) τότε δημιουργούμε ένα νήμα με παράμετρο την συνάρτηση για την αποστολή μηνύματος, δίνουμε ένα νέο όνομα στο νήμα και το εκκινούμε.

Γραμμή 43-46: Θέτουμε το flag σε ψευδές, σταματάμε οποιαδήποτε αναπαραγωγή καθώς το σημείο είναι κρίσιμο και τέλος αναπαράγουμε το φωνητικό μήνυμα που έχουμε ορίσει.

```
50              if self.t1.is_alive() is False
                and self.t3.is_alive() is False:
51                  self.t1 = threading.Thread(target=Yellow_LED)
52                  self.t1.start()
53
54              if (TotalDetections % 10) not in [0,1]:
55                  self.FSoundFlag = True
56                  self.once_warning = 0
57              else:
58                  self.FSoundFlag = False
```

Γραμμή 50-52: Βεβαιωνόμαστε ότι το πράσινο και το κόκκινο LED βρίσκονται σε ψευδή κατάσταση και τότε ενεργοποιούμε το κίτρινο LED.

Γραμμή 54-58: Εάν το υπόλοιπο του μετρητή συνόλου εντοπισμών είναι ίσο με 0 ή 1 τότε θέτουμε ξανά θέτουμε νέες τιμές, αλλιώς κρατάμε το flag ψευδές ώστε να μην υπάρχει πρόβλημα.

```
61              # If pygame isn't playing an alarm then
62              if pygame.mixer.Channel(0).get_busy() == False:
63                  # If current time is greater than the alarm length
```

```
64                  # then we play the alarm again, that means that if
65                  # the alarm already play and we continue to detect
66                  # drowsiness the alarm would play after it finish
67                  if time() > self.ALARM_FINISH:
68                      #print('ALARM TRUE ON EYES!')
69                      self.ALARM_FINISH = time()+alarm.get_length()
70                      alarm.play()
71                      self.AdvisoryMessage = True
72
73          # Icrease a counter each time we found drowsiness
74          if self.EYE_COUNTER % MAX_FRAMES == 0:
75              self.TimesDrowsy += 1
```

Γραμμή 62: Βλέπουμε εάν το κανάλι 0 το οποίο αναπαράγει τους συναγερμούς, μας επιστρέφει κατάσταση ψευδής, δηλαδή δεν αναπαράγει κάποιον συναγερμό την συγκεκριμένη στιγμή.

Γραμμή 67-71: Εάν ο χρόνος είναι μεγαλύτερος από το χρόνο + διάρκεια του συναγερμού, τότε ξανά αναπαράγουμε το συναγερμό. Αυτό συμβαίνει ώστε να μην αναπαράγουμε τους συναγερμός συνεχόμενα και αυτοί παρεμβάλλονται μεταξύ τους.

Γραμμή 74-75: Εάν ο αριθμός των πλαισίων που έχουμε εντοπίσει συνεχόμενα κλειστά μάτια, φτάσει τον μέγιστο αριθμό πλαισίων που έχουμε ορίσει ως όριο, τότε αυξάνουμε τον μετρητή εντοπισμών κατά 1.

```
77      else:
78          if self.t2.is_alive() is False and self.FLAG_LED == True:
79              self.t2 = threading.Thread(target=Green_LED)
80              self.t2.start()
81              self.FLAG_LED = False
82          self.FINAL_FLAG = False
83          self.EYE_COUNTER = 0
84          self.once_warning = 0
85          self.FLAG = False
```

Γραμμή 77: Βρισκόμαστε στην περίπτωση όπου η μέση τιμή των ματιών είναι μεγαλύτερη από την τιμή κατωφλίου που έχουμε θέσει, και επομένως σε αυτήν την περίπτωση τα μάτια είναι ανοιχτά.

Γραμμή 78-81: Εάν το νήμα που αφορά το πράσινο LED δεν τρέχει ΚΑΙ το flag του LED είναι αληθές δηλαδή μπορεί να είναι ενεργοποιημένο σε διαφορετικό χρώμα, τότε ενεργοποιούμε το πράσινο LED.

Γραμμή 82-85: Εκχωρούμε τιμές στα flag και στους μετρητές με τις αρχικές τους καταστάσεις.

```
88          # ******************
89          # ADVISORY MESSAGES
90          # ******************
91          if self.AdvisoryMessage == True and (TotalDetections%10) < 5:
```

```
92              if pygame.mixer.Channel(0).get_busy() == False:
93                  # GENERAL MESSAGES
94                  n = random.randint(1,6)
95                  advisory =
                    pygame.mixer.Sound
                    ("AdvisoryMessages/General/"+str(n)+".mp3")
96                  advisory.play()
97                  self.AdvisoryMessage = False
98          elif self.AdvisoryMessage == True
            and 5 <= (TotalDetections%10) < 10:
99              if pygame.mixer.Channel(0).get_busy() == False:
100                 # WARNING MESSAGES
101                 n = random.randint(1,2)
102                 advisory =
                    pygame.mixer.Sound
                    ("AdvisoryMessages/Warning/"+str(n)+".mp3")
103                 advisory.play()
104                 self.AdvisoryMessage = False
```

Γραμμή 91-97: Εάν το flag (AdvisoryMessage) είναι αληθές ΚΑΙ ο μετρητής εντοπισμών είναι κάτω από 5 (1 έως 4) και επιπλέον το κανάλι 0 (φωνητικά μηνύματα) είναι ελεύθερο, τότε αρχικά επιλέγουμε έναν τυχαίο αριθμό από το 1-6 και με βάση αυτό αναπαράγουμε ένα γενικό συμβουλευτικό μήνυμα που βρίσκεται μέσα στον κατάλογο με τα 6 γενικά συμβουλευτικά μηνύματα (Κατάλογος General) που διαθέτουμε.

Γραμμή 98-104: Αλλιώς εάν το flag (AdvisoryMessage) είναι αληθές ΚΑΙ ο μετρητής εντοπισμών είναι από 5 έως 9 και επιπλέον το κανάλι 0 (φωνητικά μηνύματα) είναι ελεύθερο, τότε αρχικά επιλέγουμε έναν τυχαίο αριθμό από το 1-2 και με βάση αυτό αναπαράγουμε ένα προειδοποιητικό μήνυμα που βρίσκεται μέσα στον κατάλογο με τα 2 προειδοποιητικά μηνύματα (Κατάλογος Warning) που διαθέτουμε.

- **LipsDetection.py**

  - Συνάρτηση **lips_aspect_ratio**:

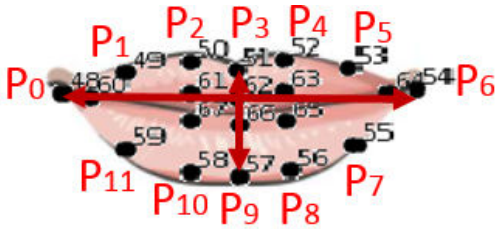
```
'''
Calculates the aspect ratio from 6 points of lips
and return it
'''
def lips_aspect_ratio(self, lips):
    # Vertical distance
    A = np.linalg.norm(lips[2] - lips[10])

    # Horizontal distance
    B = np.linalg.norm(lips[4] - lips[8])

    # Horizontal distance
    B = np.linalg.norm(lips[0] - lips[6])

    lar = float( (A/B) / (2.0 * C))
    return lar
```

Παράμετροι:

lips – 6 σημεία(x,y) τα οποία βρίσκονται γύρω από τα χείλη (Εικόνα 7.3 (α))

Επεξήγηση συνάρτησης:

Η συνάρτηση όπως βλέπουμε αρχικά βρίσκει τη διαφορά της απόστασης, μεταξύ των σημείων των χειλιών που έχουμε προσδιορίσει. Ύστερα χρησιμοποιώντας τον τύπο LAR (Lips Aspect Ratio) που βλέπουμε στην Εικόνα 7.3 (β), υπολογίζει και επιστρέφει τη διαφορά της απόστασης των χειλιών, δηλαδή όσο πιο ανοιχτά είναι τα χείλη, τόσο μεγαλύτερη θα είναι η τιμή, ενώ όσο πιο κλειστά είναι τα χείλη, τόσο μικρότερη θα είναι η τιμή καθώς η δύο κάθετες αποστάσεις (P2-P10 & P4-P8) που υπολογίζουμε τείνουν να έρχονται πιο κοντά μεταξύ τους πλησιάζοντας το οριζόντιο σημείο (P0-P6). Τέλος επιστρέφουμε το αποτέλεσμα.



(α)

$$LAR = \frac{(P_2 - P_{10}) + (P_4 - P_8)}{2 * (P_0 - P_6)}$$

(β)

Εικόνα 7.3 (α) Τα 11 σημεία των χειλιών, (β) Τύπος για τον υπολογισμό της μέσης απόστασης των χειλιών

- Συνάρτηση **detect_yawn**:

Αυτή η συνάρτηση έχει μικρές διαφορές σε σχέση με την συνάρτηση **'eye_aspect_ratio'** που βρίσκεται στο αρχείο **'EyesDetection.py'**. Επομένως θα δούμε και θα εξηγήσουμε μόνο τα σημεία που διαφέρουν μεταξύ τους. Για την αναλυτική δομή της συνάρτησης ανατρέξτε στο κεφάλαιο 6.3, αρχείο 'EyesDetection.py', συνάρτηση **'eye_aspect_ratio'.**

Παρακάτω βλέπουμε τα σημεία τα οποία έχουν αλλάξει:

```
def detect_yawn(self, frame, LipsLAR, LIPS_THRESHOLD, MAX_FRAMES, alarm, TotalDetections, FINAL_FLAG):
    '''Detect Yawn

    Args:
        frame       : Frame to process
        LipsLAR     : Lips Aspect Ratio(L.A.R)
        LIPS_THRESHOLD  : The max threshold for L.A.R
        MAX_FRAMES      : Max Frames for yawn
       alarm           : Alarm sound
        TotalDetections : Counter for total time detections
        FINAL_FLAG      : True when YAWN detected
    '''
```

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία.

LipsLAR: Η τιμή που προκύπτει από την συνάρτηση **'lips_aspect_ratio'** όπου υπολογίζει την διάσταση μεταξύ των χειλιών.

LIPS_THRESHOLD: Η τιμή κατωφλίου των χειλιών.

MAX_FRAMES: Μέγιστος αριθμός πλαισίων τον οποίον αν ξεπεράσουμε εντοπίζουμε χασμουρητό.

alarm: Ο συναγερμός που θέτουμε.

TotalDetections: Συνολικός μετρητής εντοπισμών.

FINAL_FLAG: Flag το οποίο γίνεται αληθές μόλις εντοπίσουμε χασμουρητό.

```
13      # If Lips Aspect Ratio(L.A.R) is less than Lips threshold
14      # then increase Yawn counter
15      if LipsLAR > LIPS_THRESHOLD:
16          self.FLAG = True
17          self.YAWN_COUNTER += 1
18          print("YAWN COUNTER FRAMES: ", self.YAWN_COUNTER)
19          # If the eyes remain closed for too many number of frames
20          # then we have detected a drowsiness
21          if self.YAWN_COUNTER >= MAX_FRAMES:
22              self.FLAG_LED = True
23              cv2.putText(frame, "DROWSINESS ALERT![YAWN]", (10, 30),
24              cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

Γραμμή 15: Η διαφορά με την συνάρτηση **'eye_aspect_ratio (Γραμμή 17)'** είναι ότι σε αυτήν την περίπτωση βλέπουμε εάν η διάσταση των χειλιών (LipsLAR) είναι μεγαλύτερη από το κατώφλι που έχουμε θέσει ως όριο.

Γραμμή 16-24: Οι γραμμές έχουν την ίδια λογική με τις γραμμές που βρίσκονται στην συνάρτηση **'eye_aspect_ratio (Γραμμή 15 έως 27)'**

Όλες οι υπόλοιπες γραμμές (ΕΚΤΟΣ 42,74,83): Ο κώδικας παραμένει ίδιος με βάση τον κώδικα που βρίσκεται στην συνάρτηση **'eye_aspect_ratio (Γραμμή 30 έως 103)',** εκτός από την **γραμμή 42** που αντί να θέσουμε το όνομα στο νήμα **self.send_mail.name = "~~SendMailEyes~~"** θέτουμε το αντίστοιχο όνομα με βάση την κλάση και αλλάζει σε **self.send_mail.name = "SendMailLips"**, την **γραμμή 74** που αντί για την μεταβλητή **~~self.EYE_COUNTER~~** χρησιμοποιούμε αντίστοιχα την μεταβλητή **self.YAWN_COUNTER** και την **γραμμή 83** που αντί να εκχωρούμε την τιμή 0 στην μεταβλητή **~~self.EYE_COUNTER=0~~**, εκχωρούμε την τιμή 0 στην αντίστοιχη μεταβλητή **self.YAWN_COUNTER=0.**

- **ArmHeadDetection.py**

- Μέθοδος **__init__**:

Η μέθοδος __init__ χρησιμοποιείται έτσι ώστε να αρχικοποιούμε μεταβλητές, να δημιουργούμε αντικείμενα τα οποία θα χρησιμοποιήσουμε μέσα στην κλάση που χρησιμοποιούμε, στην προκειμένη περίπτωση στην κλάση ArmHeadDetection.py**.**

Παρακάτω θα δούμε τις παραμέτρους της μεθόδου __init__ όπου χρησιμοποιήσαμε ώστε δημιουργήσουμε ένα αντικειμένου μέσω του framework mediapipe, ώστε να το χρησιμοποιούμε στις παρακάτω μεθόδους της κλάσης (Δεν θα δείξουμε όλες τις μεταβλητές που βρίσκονται στην μέθοδο __init__ καθώς μπορούμε να τις βρούμε στο αρχείο ArmHeadDetection.py).

```
def __init__(self, mode=False, modelComp=1, smooth=True,
               detectionConf=0.5, trackConf=0.5):
        self.mode = mode
        self.modelComp = modelComp
        self.smooth = smooth
        self.detectionConf = detectionConf
        self.trackConf = trackConf
        self.mpDraw = mp.solutions.drawing_utils
        self.mpPose = mp.solutions.pose
        self.pose = self.mpPose.Pose
        (self.mode,  self.modelComp,
         self.smooth, self.detectionConf,
         self.trackConf)
```

Παράμετροι:

mode: Έχουμε ορίσει την τιμή προεπιλογής να είναι **ψευδής**, αυτό σημαίνει ότι επεξεργαζόμαστε τα πλαίσια σε πραγματικό χρόνο ή έχουμε ως είσοδο μια ροή βίντεο. Αυτή η επιλογή ανιχνεύει το άτομο από τις πρώτες εικόνες, βρίσκει τις συντεταγμένες των σημείων πάνω στο σώμα του ατόμου, και ύστερα παρακολουθεί τα σημεία αυτά. Στην περίπτωση που ορίσουμε την τιμή ως **αληθής** τότε ο εντοπισμός του ατόμου και όλοι η παραπάνω διαδικασία εκτελείται ξεχωριστά για κάθε πλαίσιο εισόδου.

modeComp: Έχουμε ορίσει την τιμή προεπιλογής να είναι **1,** το οποίο αφορά την ακρίβεια των σημείων εντοπισμού. Οι επιλογές που έχουμε είναι **0, 1, 2** όσο μεγαλύτερη η τιμή τόσο μεγαλύτερη πολυπλοκότητα και επομένως αυξημένη καθυστέρηση επεξεργασίας.

smooth: Έχουμε ορίσει την τιμή προεπιλογής να είναι **αληθής** και αυτό που κάνει η επιλογή μας είναι κατά την διάρκεια επεξεργασίας του πλαισίου να μειώσει το τρέμουλο (jitter) που δημιουργείται σε μια εικόνα.

detectionConf: Έχουμε ορίσει την τιμή προεπιλογής να είναι **0.5,** καθορίζοντας έτσι ότι η ελάχιστη ακρίβεια για το εντοπισμό ενός ατόμου στο πλαίσιο, θα είναι κάπου στην μέση. Οι τιμές που μπορούμε να δώσουμε κυμαίνονται από **0.0 έως 1.0** ανάλογα με τον βαθμό ακρίβειας που θέλουμε να έχουμε, ώστε να εντοπίσουμε ένα άτομο στο πλαίσιο μας.

trackConf: Έχουμε ορίσει την τιμή προεπιλογής να είναι **0.5,** καθορίζοντας έτσι ότι η ελάχιστη ακρίβεια για την παρακολούθηση των σημείων πάνω στο σώμα του άτομο, θα είναι κάπου στην μέση. Οι τιμές που μπορούμε να δώσουμε κυμαίνονται από **0.0 έως 1.0** ανάλογα με τον βαθμό ακρίβειας που θέλουμε να έχουμε, ώστε να παρακολουθούμε τα σημεία πάνω στο σώμα του ατόμου.

Γραμμή 3-7: Εκχωρούμε τις τιμές που αναφέραμε παραπάνω, στις αντίστοιχες μεταβλητές.

Γραμμή 8: Δημιουργούμε αντικείμενο το οποίο μας δίνει, βοηθητικές μεθόδους για την γραφική αναπαράσταση.

Γραμμή 9: Δημιουργούμε αντικείμενο το οποίο μας δίνει, το βασικό σκέλος για τον εντοπισμό των σημείων του σώματος.

Γραμμή 10: Δημιουργούμε το βασικό αντικείμενο που θα χρησιμοποιήσουμε στο κύριο μέρος του προγράμματος μας, περνώντας ως παραμέτρους τα χαρακτηριστικά που είδαμε παραπάνω.

- Συνάρτηση **findPose**:

```
1 '''
2 Convert the color of frame and call the model of MediaPipe
3 to find the body points
4 '''
5 def findPose(self, frame, draw=True):
6     imgRGB = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
7     self.results = self.pose.process(imgRGB)
8     if self.results.pose_landmarks:
9         # Draw the landmakrs of body points
10        if draw:
11              self.mpDraw.draw_landmarks
                (frame,
                 self.results.pose_landmarks,
                 self.mpPose.POSE_CONNECTIONS)
12        return frame
```

Χρήση συνάρτησης:

Η συνάρτηση εντοπίζει το άτομο και βρίσκει τα σημεία τους σώματος του.

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία

draw: Λαμβάνει 2 τιμές (True=Αληθής ή False=Ψευδής) με προεπιλεγμένη τιμή **αληθής**. Αφήνουμε την τιμή **True** εάν θέλουμε να εμφανίζουμε γραφικά τα σημεία του σώματος που έχουμε εντοπίζει στο πλαίσιο που εμφανίζεται όταν τρέχουμε το πρόγραμμα μας.

Επεξήγηση συνάρτησης:

Γραμμή 6: Όπως γνωρίζουμε το κάθε pixel του πλαισίου αποτελείται από βασικά 3 χρώματα το κόκκινο, πράσινο, μπλε (RGB). Αντίθετα το πλαίσιο μου λαμβάνουμε ως παράμετρο, αποτελείται από αυτά τα χρώματα, αλλά τα pixel του βρίσκονται σε αντίθετη φορά, δηλαδή μπλε, πράσινο, κόκκινο (BGR). Για να μπορέσουμε λοιπόν να επεξεργαστούμε και να χρησιμοποιήσουμε τις περαιτέρω συναρτήσεις πάνω στο πλαίσιο μας, **μετατρέπουμε τα pixel του πλαισίου από BGR σε RBG.**

Γραμμή 7: Η μέθοδος **process** αρχικά εντοπίζει το άτομο στο τρέχον πλαίσιο, ύστερα εντοπίζει τα σημεία του σώματος του.

Γραμμή 8: Εάν υπάρχουν αποτελέσματα (δηλαδή εντοπίστηκε το άτομο και τα σημεία του), τότε συνεχίζουμε.

Γραμμή 9-12: Εάν η τιμή του **draw** είναι **αληθής** όπως η προεπιλεγμένη, τότε συνεχίζουμε και καλούμε την μέθοδο **draw_landmarks** η οποία αναπαριστά γραφικά τα σημεία που βρέθηκαν, πάνω στο πλαίσιο που έχουμε εισάγει. Τέλος επιστέφουμε αυτό το πλαίσιο με την γραφική αναπαράσταση.

- Συνάρτηση **getPosition**:

```
'''
Take a frame and return a list, with id of every body point and his
coordinates
'''
1 def getPosition(self, frame):
2     self.lmList = []
3     if self.results.pose_landmarks:
4         for id, lm in enumerate(self.results.pose_landmarks.landmark):
5             h, w, _ = frame.shape
6             cx, cy = int(lm.x * w), int(lm.y * h)
7             self.lmList.append([id, cx, cy])
8     return self.lmList
```

Χρήση συνάρτησης:

Η συνάρτηση με βάση τα σημεία του σώματος που έχουν εντοπισθεί, δημιουργεί και επιστρέφει μια λίστα, με τις συντεταγμένες των σημείων του σώματος.

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία.

Γραμμή 2-3: Δημιουργούμε μια λίστα και έπειτα ελέγχουμε αν υπάρχουν διαθέσιμα αποτελέσματα.

Γραμμή 4: Εκτελούμε μια επανάληψη για κάθε σημείο (landmark) που βρίσκεται μέσα στα αποτελέσματα και το απαριθμούμε μέσω της συνάρτηση **enumerate**.

Γραμμή 5: Η μέθοδος **shape** μας επιστρέφει το μέγεθος του πλαισίου.

Γραμμή 6-9: Βρίσκουμε τις συντεταγμένες των σημείων του σώματος (cx, xy) πάνω στο πλαίσιο, και τις προσθέτουμε σε μια λίστα απαριθμημένες (μεταβλητή id). Τέλος επιστρέφουμε την λίστα.

- Συνάρτηση **findAngle**:

```
'''
Give 3 points of body and return an angle between them
(We used this function to refer to 3 points of left and right arm)
'''
1  def findAngle(self, frame, p1, p2, p3, draw=True):
2
3      # Get the coordinates for points p1, p2, p3
4      x1, y1 = self.lmList[p1][1:]
5      x2, y2 = self.lmList[p2][1:]
6      x3, y3 = self.lmList[p3][1:]
7
8      # Calculate the angle
9      angle = math.degrees(
       math.atan2(y3-y2, x3-x2) - math.atan2(y1-y2, x1-x2))
10
11     if angle < 0:
12         angle += 360
13
14     # Draw the body points p1, p2, p3 and the lines between them
15     if draw:
16         cv2.line(frame, (x1, y1), (x2, y2), (255, 255, 255), 2)
17         cv2.line(frame, (x2, y2), (x3, y3), (255, 255, 255), 2)
18         cv2.circle(frame, (x1, y1), 8, (0, 0, 255), cv2.FILLED)
19         cv2.circle(frame, (x1, y1), 10, (0, 0, 255), 1)
20         cv2.circle(frame, (x2, y2), 8, (0, 0, 255), cv2.FILLED)
21         cv2.circle(frame, (x2, y2), 10, (0, 0, 255), 1)
22         cv2.circle(frame, (x3, y3), 8, (0, 0, 255), cv2.FILLED)
23         cv2.circle(frame, (x3, y3), 10, (0, 0, 255), 1)
24         cv2.putText(frame, f"ANGLE: {str(int(angle))}",
           (x2 - 210, y2 + 20),
25                     cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 255), 2)
26     return int(angle)
```

Χρήση συνάρτησης:

Η συνάρτηση λαμβάνει ως παραμέτρους 3 σημεία του σώματος (p1, p2, p3) και βρίσκει τις μοίρες μιας γωνίας μεταξύ των σημείων. Εμείς χρησιμοποιούμε αυτήν την συνάρτηση

για να βρούμε τις μοίρες της γωνίας του αριστερού και του δεξιού χεριού. π.χ. οι τιμές των παραμέτρων ώστε να βρούμε τις μοίρες του δεξιού χεριού θα είναι p1=11 (αριστερός ώμος), p2=13 (αριστερός αγκώνας) και p3=15 (αριστερός καρπός).

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία.

p1, p2, p3: 3 σημεία του σώματος από τα οποία θα βρούμε την γωνία τους.

draw: Λαμβάνει 2 τιμές (True=Αληθής ή False=Ψευδής) με προεπιλεγμένη τιμή **αληθής**. Αφήνουμε την τιμή **True** εάν θέλουμε να αναπαραστήσουμε γραφικά στο πλαίσιο, την ένωση των 3 σημείων αυτών μεταξύ 2 γραμμών, και την εμφάνιση των μοιρών της γωνίας που σχηματίζουν.

Γραμμή 4-6: Κρατάμε στις αντίστοιχες μεταβλητές (x1,y1 κτλ.), τις συντεταγμένες των 3 σημείων του σώματος που βρίσκονται στο πλαίσιο.

Γραμμή 9: Βρίσκουμε τις μοίρες της γωνίας των 3 σημείων(p1, p2, p3), χρησιμοποιώντας την μαθηματική συνάρτηση degrees.

Γραμμή 11: Εάν το αποτέλεσμα των μοιρών είναι αρνητικό τότε προσθέτουμε 360 μοίρες ώστε να γίνει θετικό.

Γραμμή 15-26: Εφόσον η τιμή **draw** είναι **αληθής**, τότε δημιουργούμε 2 γραμμές που ενώνουν τα 3 σημεία μεταξύ τους, για κάθε ένα από τα 3 σημεία δημιουργούμε έναν κύκλο και τον γεμίζουμε με κόκκινο χρώμα και έπειτα εμφανίζουμε τις μοίρες στο πλαίσιο δίπλα από τον αγκώνα (Εικόνα 7.4). Τέλος επιστρέφουμε το ακέραιο αποτέλεσμα των μοιρών.

Εικόνα 7.4 Αριστερό χέρι σε γωνία 38 μοιρών

- Συνάρτηση **hand_ear_aspect_ratio**:

```
'''
Find the distance between points p1 and p2
(We used this function to find the dinstace between the hand and ear of
driver)
'''
def hand_ear_aspect_ratio(self, frame, p1, p2, draw=True, Distance=0):

    # Get the coordinates for points p1, p2
    x1, y1 = self.lmList[p1][1:]
    x2, y2 = self.lmList[p2][1:]

    a = np.array([x1, y1])
    b = np.array([x2, y2])

    # Find the distance
    distance = np.linalg.norm(a - b)

    # We use this for tilted head
    if Distance > 0:
        distance = (distance+Distance)/2

    # Draw the body points p1, p2
    if draw:
        cv2.circle(frame, (x1, y1), 5, (0, 0, 255), cv2.FILLED)
        cv2.circle(frame, (x1, y1), 10, (0, 0, 255), 1)
        cv2.circle(frame, (x2, y2), 5, (0, 0, 255), cv2.FILLED)
        cv2.circle(frame, (x2, y2), 10, (0, 0, 255), 1)

    return int(distance)
```

Χρήση συνάρτησης: Η συνάρτηση βρίσκει την απόσταση μεταξύ των 2 σημείων (p1, p2) που έχουμε ορίσει. Χρησιμοποιούμε αυτή την συνάρτηση ώστε να βρίσκουμε την απόσταση μεταξύ χεριού και αυτιού του οδηγού, με σκοπό να μπορούμε να εντοπίσουμε πιθανόν χρήση κινητού τηλεφώνου ή περίεργη συμπεριφορά του οδηγού. Μια ακόμη χρήση της συνάρτησης είναι για την εύρεση την απόστασης μεταξύ του αυτιού και του ώμου, με σκοπό τον εντοπισμό πιθανής κούρασης ή περίεργης συμπεριφοράς.

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία.

p1, p2: 2 σημεία από τα οποία βρίσκουμε την απόσταση.

draw: Λαμβάνει 2 τιμές (True=Αληθής ή False=Ψευδής) με προεπιλεγμένη τιμή **αληθής**. Αφήνουμε την τιμή **True** εάν θέλουμε να εμφανίσουμε στο πλαίσιο την γραφική αναπαράσταση των 2 σημείων.

Distance: Η απόσταση, του προσώπου του οδηγού από την κάμερα.

Γραμμή 4-5: Κρατάμε στις αντίστοιχες μεταβλητές (x1,y1 κτλ.), τις συντεταγμένες των 2 σημείων.

Γραμμή 7-8: Μετατρέπουμε τις μεταβλητές των συντεταγμένων σε numpy πίνακες, με χρήση της βιβλιοθήκης NumPy.

Γραμμή 11: Βρίσκουμε την απόσταση των 2 σημείων.

Γραμμή 14-15: Χρησιμοποιείται για να υπολογίσει την κλίση του κεφαλιού ως προς τον ώμο.

Γραμμή 18-24: Αναπαριστά γραφικά τα 2 σημεία που έχουμε ορίσει με 2 κόκκινους κύκλους. Τέλος επιστρέφουμε την ακέραια απόσταση των 2 σημείων.

- Συνάρτηση **detect_phone_hand**:

Η δομή της συνάρτησης είναι ίδια με την συνάρτηση **'eye_aspect_ratio'** που είδαμε παραπάνω**.** Η συνάρτηση χρησιμοποιείται για τον εντοπισμό χρήσης κινητού τηλεφώνου ή τον εντοπισμό του χεριού σε μη κανονική θέση.

```
1   def detect_phone_hand(self, frame, HEAR, HEAR_THRESHOLD,
                          MAX_FRAMES, alarm, TotalDetections,
                          WARNING_BOTH_HANDS=False,FINAL_FLAG=False,
                          choose_operator='less'):
2
3
4       '''Possibility of cell phone detection OR hand detection
        in incorrect position
        (This function used for ear to hand aspect ratio)
5
6
7       Args:
8           frame              : Frame to process
9           HEAR               : Hand Ear Aspect Ratio(H.E.A.R)
10          HEAR_THRESHOLD     : The max threshold for H.E.A.R
11          MAX_FRAMES         : Max Frames for HEAR
12          alarm              : Play an alarm sound
13          TotalDetections    : Counter for total time detections
14          WARNING_BOTH_HANDS :
            Flag tha used when we detected strange behaviour for each arm
15          FINAL_FLAG         :
            True when phone or strange behaviour detected
16          operator           : less : '<' and greater : '>'
17      '''
```

Παράμετροι:

frame: Τρέχον πλαίσιο για επεξεργασία.

HEAR: Η τιμή της απόσταση μεταξύ αυτιού και χεριού.

HEAR_THRESHOLD: Η τιμή κατωφλίου της απόστασης του HEAR.

MAX_FRAMES: Μέγιστος αριθμός πλαισίων τον οποίον αν ξεπεράσουμε εντοπίζουμε περίεργη συμπεριφορά.

alarm: Ο συναγερμός που θέτουμε.

TotalDetections: Συνολικός μετρητής εντοπισμών.

WARNING_BOTH_HANDS: Flag το οποίο γίνεται **'Αληθές',** όταν έχουμε εντοπίσει και για τα 2 χέρια περίεργη συμπεριφορά.

FINAL_FLAG: Flag το οποίο γίνεται αληθές μόλις εντοπίσουμε χασμουρητό

```
18      # If Hand Ear Aspect Ratio(H.E.A.R) is less than HEAR threshold
19      # then increase HEAR counter
20      if HEAR < HEAR_THRESHOLD:
```

Γραμμή 20: Βλέπουμε εάν η απόσταση μεταξύ αυτιού και χεριού, είναι μικρότερη από την τιμή κατωφλίου που έχουμε θέσει.

Όλες οι υπόλοιπες γραμμές: Η λογική του κώδικα παραμένει ίδια με βάση τον κώδικα που βρίσκεται στην συνάρτηση **'eye_aspect_ratio',** με μόνη διαφορά την χρήση διαφορετικών μεταβλητών.

- Συνάρτηση **both_hands_detect**:

Η συνάρτηση χρησιμοποιείται όταν έχουμε εντοπίσει μη κανονική θέση και στα δύο χέρια, και εκτελεί λειτουργίες που έχουμε ήδη περιγράψει.

- Συνάρτηση **detect_incorrect_right_arm**:

Η διαφορά με την συνάρτηση **'detect_phone_hand'** βρίσκεται στις 3 παραμέτρους (HEAR, HEAR_THRESHOLD, MAX_FRAMES) και τους ελέγχους που πραγματοποιούνται.

Παράμετροι (Ίδιοι εκτός από τους 3 παρακάτω)**:**

RIGHT_INCOR: Δέχεται έναν πίνακα δυο τιμών. Η $1^{η}$ τιμή αφορά τον αριθμό των μοιρών που για το δεξί χέρι σε μια κλίμακα από το 0 έως το 100, και η $2^{η}$ τιμή είναι ο πραγματικός αριθμός των μοιρών που βρήκαμε για το δεξί χέρι.

RIGHT_THRESHOLD: Δέχεται έναν πίνακα δυο τιμών. Η $1^{η}$ τιμή αφορά την τιμή του κατωφλίου με βάση την $1^{η}$ τιμή του HEAR. Η $2^{η}$ τιμή είναι ένα πίνακας δύο τιμών, το εύρος των οποίων αποτελούν τιμή κατωφλίου με βάση την $2^{η}$ τιμή του HEAR.

MAX_FRAMES: Δέχεται έναν πίνακα δυο τιμών. $1^{η}$ και $2^{η}$ τιμή αφορούν το μέγιστο αριθμός πλαισίων τον οποίον αν ξεπεράσουμε εντοπίζουμε περίεργη συμπεριφορά.

- Συνάρτηση **detect_incorrect_left_arm**:

Η διαφορά με την συνάρτηση **'detect_phone_hand'** βρίσκεται στις 3 παραμέτρους (HEAR, HEAR_THRESHOLD, MAX_FRAMES) και τους ελέγχους που πραγματοποιούνται.

Παράμετροι (Ίδιοι εκτός από τους 2 παρακάτω)**:**

LEFT_INCOR: Η τιμή των μοιρών της γωνίας για το αριστερό χέρι.

LEFT_THRESHOLD: Δέχεται έναν πίνακα δυο τιμών. Η $1^{η}$ τιμή αφορά έναν πίνακα δύο τιμών, το εύρος των οποίων αποτελούν την χαμηλή τιμή κατωφλίου με βάση την τιμή του HEAR. Η $2^{η}$ τιμή αντίστοιχα αφορά έναν πίνακα δύο τιμών, το εύρος των οποίων αποτελούν την υψηλή τιμή κατωφλίου με βάση την τιμή του HEAR.

➢ **smtp.py**

- Συνάρτηση **mail_format**:

```
1 def mail_format(driver_name, send_mail, send_pass,
                  recv_mail, latitude=0, longitude=0):
2
3     '''Used to send a mail in proper format
4
5         Args:
6             driver_name : Driver's name
7             send_mail   : Sender email
8             send_pass   : Sender pass for the email
9             recv_mail   : Receiver email
10            latitude    : Latitude
11            longitude   : Longitude
12    '''
```

Παράμετροι:

driver_name: Το όνομα του οδηγού.

send_email: Το email του αποστολέα.

send_pass: Ο κωδικός του email του αποστολέα.

recv_email: Το email του παραλήπτη.

latitude: Γεωγραφικό πλάτος.

longitude: Γεωγραφικό μήκος.

```
14    country = city = village = road = ""
15
16    # Base URL to Nomitanim
17    url_start
      = "https://nominatim.openstreetmap.org/reverse?format=json&"
18    url_end = "&zoom=18&addressdetails=1&accept-language=en"
19
20    try:
```

```
21          if latitude != 0 and longitude != 0:
22              # GET Response
23              url_json = url_start + "lat=" + str(latitude) +
                "&lon=" + str(longitude) + url_end
24              r = requests.get(url_json)
25
26              # URL that displays the actual driver's location
                # on the map (Send via email)
27              map_url
                = "https://nominatim.openstreetmap.org/ui/reverse.html?" \
28                        + "lat=" + str(latitude) \
29                        + "&lon=" + str(longitude) \
30                        + "&zoom=16"
31
32              # Keep dictionary with information about address
33              address = r.json()['address']
```

Γραμμή 17-18: Αρχή και τέλος του URL ώστε να λαμβάνουμε τα δεδομένα μας σε σωστή μορφή από το **nominatim.openstreetmap**.

Γραμμή 20-24: Αφού υπάρχουν τιμές για το μήκος (longitude) και πλάτος (latitude), συνεχίζουμε και εκτελούμε ένα **HTTP GET REQUEST** και εκχωρούμε τα αποτελέσματα μας στην μεταβλητή **r.**

Γραμμή 26-30: Δημιουργούμε την κατάλληλη δομή του URL με τις συντεταγμένες, έτσι ώστε όταν στείλουμε το URL στον παραλήπτη, να μπορεί να ανοίξει το URL και να δει ακριβώς την τοποθεσία του οδηγού πάνω στο χάρτη.

Γραμμή 33: Κρατάμε σε ένα λεξικό(dictionary) τα στοιχεία της διεύθυνσης.

```
36              # Keep some info about country, city and road
37              if 'country' in address:
38                  country = address['country']
39              if 'city' in address:
40                  city = address['city']
41              if 'road' in address:
42                  road = address['road']
43              if 'village' in address:
44                  village = address['village']
```

Γραμμή 36-44: Κρατάμε ξεχωριστά σε μεταβλητές τα στοιχεία (αν υπάρχουν) της χώρας, πόλης, δρόμου και του χωριού.

```
46          # Create message object instance
47          msg = MIMEMultipart()
48
49          # Setup the parameters for the message
50          msg['From'] = str(send_mail)
51          msg['To'] = str(recv_mail)
52          msg['Subject'] = f"{driver_name}, Detected Tired!"
53
54          # Send the location if we have it
55          if latitude == 0 or longitude == 0:
56                      message = (f"<html><head></head>"
```

```
                        f"<body>"
                        f"<h3>{driver_name}, was detected tired!</h3>"
                        f"<h4>Information about
                          {driver_name} location</h4>"
                        f"<h4>LOCATION: UNKNOWN </h4>"
                        f"</body></html>")

57          else:
58              message = (f"<html><head></head>"
59                         f"<body>"
60                         f"<h3>{driver_name}, was detected tired!</h3>"
61                         f"<h4>Information about
                           {driver_name} location</h4>"
62                         f"<p>Country  :
                           {country if country else 'Unknown'}<br>"
63                         f"{f'City      : {city}<br>' if city else f''}"
64                         f"{f'Village   : {village}<br>' if village else f''}"
65                         f"Road        : {road if road else
'Unknown'}</p>"
66                         f"<b>Click in the link to see his location:
</b>"
67                         f"{map_url}<br>"
68                         f"</body></html>")
```

Γραμμή 47: Δημιουργούμε ένα αντικείμενο MIMEMultipart το οποίο θα μας βοηθήσει να στήσουμε το μήνυμα μας.

Γραμμή 49-52: Σετάρουμε το email του αποστολέα, το email του παραλήπτη και τον τίτλο του μηνύματος.

Γραμμή 54-56: Εάν δεν έχουμε λάβει τιμές για το γεωγραφικό μήκος και πλάτος, σετάρουμε το μήνυμα με άγνωστη τοποθεσία.

Γραμμή 57-68: Αλλιώς σετάρουμε το μήνυμα με όσες περισσότερες πληροφορίες έχουμε.

```
70          # Add in the message body
            msg.attach(MIMEText(message, 'html'))
75
76          # Set the mail server
77          server = smtplib.SMTP('smtp.gmail.com: 587')
78
79          # Starts an encrypted connection
80          server.starttls()
81
82          # Login Credentials
83          server.login(send_mail, send_pass)
84
85          # Send the message
86          server.sendmail(send_mail, recv_mail, msg.as_string())
87          server.quit()

88          # Enable GPS if it's disabled
89          Check_and_Enable_GPS()
```

Γραμμή 70-74: Πακετάρουμε το μήνυμα σε html μορφή.

Γραμμή 77: Σετάρουμε την κατάλληλη πόρτα για επικοινωνία με gmail SMTP Server.

Γραμμή 80: Αρχίζουμε μια κρυπτογραφημένη επικοινωνία.

Γραμμή 83: Πραγματοποιούμε την σύνδεση.

Γραμμή 86-87: Στέλνουμε το email και κλείνουμε την επικοινωνία με τον SMTP server.

Γραμμή 88-89: Καλούμε μια συνάρτηση η όποια θα ενεργοποιήσει το GPS ξανά σε περίπτωση που έκλεισε για να ενεργοποιηθεί το GSM.

```
91      except requests.exceptions.RequestException:
92          print('An error has occurred while trying to request')
93      except smtplib.SMTPResponseException:
94          print('An error has occurred while trying to send an email!')
95      except BaseException as error:
96          print('An error has occurred and email was not send it!
                \nCaught this error: ' + repr(error))
```

Γραμμή 91-96: Εξαιρέσεις οι οποίες μας βοηθούν, να εντοπίσουμε συγκεκριμένα σφάλματα, σε περίπτωση σφάλματος κατά την αποστολή του μηνύματος.

- **gps.py**

- Συνάρτηση **Get_GPS_Coordinates**:

Χρησιμοποιούμε την συνάρτηση ώστε το A9G Module όταν βρίσκεται σε λειτουργία GPS, να μας επιστρέψει το γεωγραφικό πλάτος και μήκος (Εικόνα 7.5).

```
1  def Get_GPS_Coordinates():
2
3      # Initialize the serial port
4      port="/dev/ttyS0"
5      ser=serial.Serial(port, baudrate=9600, timeout=1)
6
7      try:
8          lat = 0
9          long = 0
10
11         ser.write(str.encode('AT+LOCATION=2\r'))
12         sleep(10)
13
14         newdata = ser.readline().decode("utf-8").strip()
15         sleep(1)
16         # Read the GPS data
17         newdata = ser.readline()
18
19         # Split the data
20         newdata = newdata.split(b',')
21
22         lat  = newdata[0].decode("utf-8").strip()
23         long = newdata[1].decode("utf-8").strip()
24
25         # Flush the buffer
26         ser.flushOutput()
27
28         return True,lat,long
```

```
29
30      except:
31          print('An error has occurred while trying to get the GPS coordinates!')
32          return False,0,0
```

Γραμμή 5: Θέτουμε την σειριακή πόρτα επικοινωνίας και έπειτα δημιουργούμε ένα αντικείμενο το οποίο θα χρησιμοποιούμε για την επικοινωνία με το A9G Module. Οι παράμετροι που χρησιμοποιούμε είναι η σειριακή πόρτα επικοινωνίας (port), ο μέγιστος αριθμός μετάδοσης bits ανά δευτερόλεπτο (baudrate=9600) και το χρονικό όριο για ανάγνωση (timeout=1).

Γραμμή 11: Στέλνουμε σειριακά την εντολή **'AT+LOCATION=2'** στο A9G Module, ώστε να μας επιστρέψει το γεωγραφικό πλάτος και μήκος (Εικόνα 7.5).

Γραμμή 17: Διαβάζουμε τα δεδομένα που μας επέστρεψε η παραπάνω εντολή (γεωγραφικό πλάτος και μήκος).

Γραμμή 20: Διαχωρίζουμε τις δύο τιμές μέσω του κόμματος (',') και τις εκχωρούμε σε ένα πίνακα.

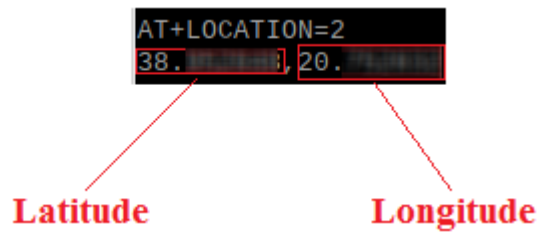
Γραμμή 22: Αποκωδικοποιούμε και κρατάμε το γεωγραφικό πλάτος σε μια μεταβλητή (lat).

Γραμμή 23: Αποκωδικοποιούμε και κρατάμε το γεωγραφικό μήκος σε μια μεταβλητή (long).

Γραμμή 26: Καθαρίζουμε τον buffer της σειριακής επικοινωνίας.

Γραμμή 28: Επιστρέφουμε την τιμή **'True'** ότι πήγαν όλα καλά, την τιμή **'lat'** για το γεωγραφικό πλάτος και την τιμή **'long'** για το γεωγραφικό μήκος.

Γραμμή 30-32: Σε περίπτωση που κατά την εκτέλεση του προγράμματος κάτι πάει λάθος ή η εντολή **'AT+LOCATION=2'** δεν μας έχει επιστρέψει τις συντεταγμένες ('+LOCATION: GPS NOT FIX NOW'), τότε επιστρέφουμε την τιμή **'False'** ότι κάτι δεν πήγε καλά, και για το γεωγραφικό πλάτος και μήκος (**'lat'** & **'long'**) θέτουμε τις τιμές 0 (όπου σε αυτήν την περίπτωση θεωρούμε πως οι συντεταγμένες δεν έχουν εντοπισθεί).



Εικόνα 7.5 Δεδομένα GPS από το A9G Module (AT+LOCATION=2)

- **gsm.py**

  - Συνάρτηση **Check_GSM_State()**:

```
1  def Check_GSM_State():
2
3      # Initialize the serial port
4      port="/dev/ttyS0"
5      ser=serial.Serial(port, baudrate=9600, timeout=1)
6
7      # Reset the A9G Module
8      LED_Initialize()
9      sleep(1)
10     Reset_HIGH_Module()
11
12     # Disable GPS if it's enabled already
13     ser.write(str.encode('AT+GPSRD=0\r'))
14     ser.write(str.encode('AT+GPS=0\r'))
15
16     # Stop the ppp script in case that already run
17     subprocess.call(["sudo", "poff", "rnet"], cwd="/etc/ppp/peers/")
```

Γραμμή 4-5: Θέτουμε την σειριακή πόρτα επικοινωνίας και έπειτα δημιουργούμε ένα αντικείμενο το οποίο θα χρησιμοποιούμε για την επικοινωνία με το A9G Module. Οι παράμετροι που χρησιμοποιούμε είναι η σειριακή πόρτα επικοινωνίας (port), ο μέγιστος αριθμός μετάδοσης bits ανά δευτερόλεπτο (baudrate=9600) και το χρονικό όριο για ανάγνωση (timeout=1).

Γραμμή 8-10: Αρχικοποιούμε τα GPIO Pins και ενεργοποιούμε το A9G Module.

Γραμμή 13-14: Στέλνουμε σειριακά την εντολή **'AT+GPSRD=0'** ώστε το A9G Module να σταματήσει να παράγει δεδομένα από GPS και την εντολή **'AT+GPS=0'** ώστε το A9G Module να απενεργοποιήσει την λειτουργία του GPS.

Γραμμή 17: Σταματάμε να τρέχουμε το ppp script σε περίπτωση που τρέχει ήδη.

```
19     try:
20         # Flush the buffer
21         ser.flushInput()
22         ser.flushOutput()
23         sleep(1)
24
25         oldtime = time()
26         AT_ans = ""
27
28         while AT_ans != "OK" and not (time() - oldtime > 60):
29             # Check if module is communicate through serial port
30             # Write an 'AT' command
31             ser.write(str.encode('AT\r'))
32             sleep(0.5)
33
34             # Receive the answer
35             ser.readline()
```

```
36              AT_ans = ser.readline().decode("utf-8").strip()
```

Γραμμή 21-22: Καθαρίζουμε τον buffer εισόδου/εξόδου της σειριακής επικοινωνίας.

Γραμμή 28: Όσο δεν λαμβάνουμε το την απάντηση **'OK'** και ο χρόνος δεν ξεπερνά το ένα λεπτό (1 λεπτό).

Γραμμή 31: Στέλνουμε την εντολή **'AT'** στο σειριακό κανάλι.

Γραμμή 35-36: Διαβάζουμε την απάντηση από το σειριακό κανάλι μέχρι να λάβουμε το **'OK'** που σημαίνει ότι υπάρχει επικοινωνία και μπορούμε να συνεχίσουμε.

```
39          if AT_ans == "OK":
40
41              # Set the PIN of SIM card
42              ser.write(str.encode('AT+CPIN=2255\r'))
43              sleep(45)
44
45              # Receive the answer
46              ser.readline()
47              GSM_ans = ser.readline().decode("utf-8").strip()
```

Γραμμή 39: Εάν η απάντηση που λάβουμε είναι **'OK'** συνεχίζουμε.

Γραμμή 42-43: Στέλνουμε το σειριακό κανάλι την AT εντολή, η οποία προσδιορίζει το PIN της SIM κάρτας μας (τροποποιούμε τον κωδικό με τον αντίστοιχο που έχουμε) και περιμένουμε για 45 δευτερόλεπτα, έως ότου πραγματοποιηθεί η σύνδεση του A9G Module μας, με το δίκτυο κινητής τηλεφωνίας.

Γραμμή 46-47: Λαμβάνουμε την απάντηση (θέλουμε να λάβουμε **'OK'**).

```
50              if GSM_ans == "OK":
51
52                  # Activate PDP context
53                  ser.write(str.encode('AT+CGACT=1,1\r'))
54                  sleep(60)
55
56
57                  # Set the PDP context
                    ser.write(str.encode
                    ('AT+CGDCONT=1,
                    "IP","internet.vodafone.gr",
                     "0.0.0.0",0,0\r'))
58                  sleep(1)
59
60                  # Flush the buffer
61                  ser.flushInput()
62                  ser.flushOutput()
```

Γραμμή 50: Εάν η απάντηση που λάβουμε είναι **'OK'** συνεχίζουμε.

Γραμμή 53-54: Ενεργοποιούμε το PDP (Packet Data Protocol) Context, το οποίο είναι ένα πρωτόκολλο υπεύθυνο για την δρομολόγηση των πακέτων μέσω του GPRS διαδικτύου. Αναμένουμε για 60 δευτερόλεπτα μέχρι να ενεργοποιηθεί το πρωτόκολλο.

Γραμμή 57: Θέτουμε τον τύπο του PDP (στην περίπτωση μας IP) και το APN (Access Point Name) του παρόχου μας (στην περίπτωση μας ο πάροχος τηλεπικοινωνιών είναι η VODAFONE και επομένως το APN της είναι το internet.vodafone.gr). Τα υπόλοιπα πεδία παραμένουν ίδια.

Γραμμή 61-62: Καθαρίζουμε τον buffer εισόδου/εξόδου της σειριακής επικοινωνίας.

```
64                  # Check the state of PDP context
65                  ser.write(str.encode('AT+CGACT?\r'))
66                  sleep(2)
67
68                  # Receive the answer
69                  ser.readline()
70                  CONNECTED_ans = ser.readline().decode("utf-8").strip()
71
72                  if CONNECTED_ans == "+CGACT: 1, 1":
73
74
75                      # Run the ppp script
76                      subprocess.call(["sudo", "pon", "rnet"], cwd="/etc/ppp/peers/")
77                      sleep(10)
```

Γραμμή 65: Στέλνουμε την συγκεκριμένη εντολή ώστε να λάβουμε την κατάσταση του PDP Context.

Γραμμή 69-70: Διαβάζουμε το αποτέλεσμα της παραπάνω εντολής.

Γραμμή 72: Εάν το PDP Context βρίσκεται σε λειτουργία τότε συνεχίζουμε.

Γραμμή 76: Τρέχουμε το ppp script με σκοπό την πρόσβαση του Raspberry Pi στο διαδίκτυο με του A9G Module.

```
79                      # Check if there is an internet connection
80                      socket.setdefaulttimeout(5)
81                      socket.socket(socket.AF_INET,
                        socket.SOCK_STREAM).connect(("8.8.8.8", 53))
82                      return True
83
84          Check_and_Enable_GPS()
85          return False
86      except:
87          Check_and_Enable_GPS()
```

Γραμμή 80-83: Δοκιμάζουμε αν υπάρχει σύνδεση και στην περίπτωση που υπάρχει συνεχίζουμε και επιστρέφουμε αληθής τιμή. Αλλιώς σε περίπτωση σφάλματος (exception συγκεκριμένα **Γραμμή 87** καλούμε την συνάρτηση για να ενεργοποιήσουμε το GPS)

Γραμμή 84: Αλλιώς εάν δεν έχει εκτελεστεί κάποια συνθήκη από τις παραπάνω που είδαμε καλούμε την συνάρτηση για να ενεργοποιήσουμε το GPS και επιστρέφουμε τιμή ψευδής.

- **main.py**

Επεξήγηση συγκεκριμένων κομματιών κώδικα από το main αρχείο:

```
19 # Initialize GPIO
20 LED_Initialize()
21 sleep(1)
22 GPIO.setwarnings(False)
23
24 # Initialize A9G Module
25 threading.Thread(target=Enable_Module).start()
26 threading.Thread(target=Reset_HIGH_Module).start()
27
28 # Enable GPS
29 threading.Thread(target=Check_and_Enable_GPS).start()
```

Γραμμή 19-21: Αρχικοποίηση GPIO Pins και απόκρυψη σφαλμάτων

Γραμμή 25-29: Ενεργοποίηση νημάτων, ώστε να ενεργοποιήσουμε το A9G Module και ύστερα να ενεργοποιήσουμε το GPS.

```
50      # Initialize the Caffe model and the face predictor model
51      net = cv2.dnn.readNetFromCaffe
       ("models/deploy.prototxt.txt",
        "models/res10_300x300_ssd_iter_140000.caffemodel")
52      face_predictor = dlib.shape_predictor
        ('models/shape_predictor_68_face_landmarks.dat')
```

Γραμμή 51: Η μέθοδος **'readNetFromCaffe'** ΄δέχεται 2 παραμέτρους. Η 1η παράμετρος αφορά την εισαγωγή του αρχείου .prototxt, όπου είναι το αρχείο που ορίζει την αρχιτεκτονική του δικτύου, και η 2η παράμετρος αφορά την εισαγωγή του αρχείου .caffemodel, όπου αφορά τα βάρη που θα χρησιμοποιήσουν τα επίπεδα του μοντέλου. Τέλος μας επιστρέφεται ένα αντικείμενο τύπου Net. [40]

Γραμμή 52: Δημιουργούμε ένα αντικείμενο καλώντας τον κατασκευαστή (constructor) της κλάσης shape_predictor, με παράμετρο το μοντέλο ανίχνευσης 68 σημείων του προσώπου (.dat).

```
54      # EYE, LIPS and HEAR threshold
55      EYE_THRESHOLD  = 0.3
56      LIPS_THRESHOLD = 0.45
57      HEAR_THRESHOLD = 55
58
59      # TAR Dynamic variable
60      TAR_THRESHOLD = 0
61
62      # Max frames for EYE, YAWN and HEAR
63      MAX_FRAMES_EYE = 4
```

```
64      MAX_FRAMES_YAWN = 2
65      MAX_FRAMES_HEAR = 10
66      MAX_FRAMES_TAR = 5
```

Γραμμή 55: Δίνουμε μια τιμή για το όριο το οποίο θέλουμε να ανιχνεύουμε αν τα μάτια είναι κλειστά. Όσο μικρότερη η τιμή, τόσο πιο κλειστά θέλουμε να είναι τα μάτια ώστε να τα εντοπίζουμε κλειστά. Δεν θέλουμε η τιμή να είναι πολύ μεγάλη, καθώς θα εντοπίζουμε κλειστά μάτια σε περιπτώσεις που δεν θα είναι.

Γραμμή 56: Δίνουμε μια τιμή για το όριο το οποίο θέλουμε να ανιχνεύσουμε αν ο οδηγός χασμουριέται. Όσο μεγαλύτερη η τιμή σε αυτήν την περίπτωση, τόσο πιο ανοιχτό θα πρέπει να είναι το στόμα του οδηγού ώστε να τα εντοπίζουμε ανοιχτό. Δεν θέλουμε η τιμή να είναι πολύ μικρή καθώς υπάρχει πιθανότητα να εντοπίζουμε τον οδηγό σε περιπτώσεις που μιλάει ή απλά έχει ελάχιστα ανοιχτά το στόμα του.

Γραμμή 57: Δίνουμε μια τιμή, για το όριο το οποίο θέλουμε να ανιχνεύσουμε αν ο οδηγός πιθανόν μιλάει στο κινητό ή έχει κάποια περίεργη συμπεριφορά. Όσο μικρότερη η τιμή του ορίου, τόσο πιο κοντά θέλουμε να είναι το χέρι του οδηγού στο αυτί του έτσι ώστε να εντοπίζουμε περίεργη συμπεριφορά του οδηγού. Δεν θέλουμε η τιμή να είναι πολύ μεγάλη καθώς υπάρχει πιθανότητα να εντοπίζουμε περίεργη συμπεριφορά του οδηγού σε καταστάσεις που δεν είναι.

Γραμμή 60: Δίνουμε μια τιμή, για το όριο το οποίο θέλουμε να ανιχνεύσουμε αν ο οδηγός βρίσκεται πιθανόν κουρασμένος και η κλίση του κεφαλιού του τείνει ως προς τον ώμο του. Η τιμή του ορίου έχει τα ίδια χαρακτηριστικά με την τη τιμή του ορίου που είδαμε παραπάνω (Γραμμή 57), με μοναδική διαφορά ότι εδώ εντοπίζουμε περίεργη συμπεριφορά, με βάση την απόσταση του αυτιού από τον ώμο.

Γραμμή 63-66: Κάθε φορά που έχουμε ξεπερνάμε την τιμή του ορίου που έχουμε θέση, ξεκινάμε και μετράμε πλαίσια ανά εικόνα. Εάν ξεπεράσουμε τα πλαίσια ανά εικόνα που έχουμε θέση σε κάθε περίπτωση, τότε εντοπίζουμε την αντίστοιχη ανίχνευση. Όπως φαίνεται έχουμε θέση τιμές ορίων ανά πλαίσιο για τα μάτια (γραμμή 63), χασμουρητό/στόμα (γραμμή 64), απόσταση χεριού από αυτί (γραμμή 65) και απόσταση ώμου από αυτί (γραμμή 66).

```
87      # Dynamic variable that caclulated for every frame
88      # and used to to keep the distance
89      Distance = 0
90
91      # Distance from camera to object(face) measured in 'cm'
92      KNOWN_DISTANCE = 50.0
```

```
93
94      # width of face measured in 'cm'
95      KNOWN_WIDTH = 14.3
96
97      # Width(pixels) in the reference image
98      IMAGE_WIDTH = 131
99
100     # Get the focal vaule
101     Focal_length = Focal_Length_Finder \
102     (KNOWN_DISTANCE, KNOWN_WIDTH, IMAGE_WIDTH)
```

Γραμμή 89: Μεταβλητή για την εκάστοτε απόστασης της κάμερας με το πρόσωπο.

Γραμμή 92: Μεταβλητή πραγματικής απόστασης της κάμερας από το πρόσωπο (μετρημένη σε εκατοστά).

Γραμμή 95: Μεταβλητή πραγματικού πλάτους κεφαλιού.

Γραμμή 98: Μεταβλητή αντίστοιχου πλάτους εικόνας.

Γραμμή 101-102: Εύρεση εστιακού μήκους με βάση τις παραπάνω μεταβλητές.

```
104     # Starts the video stream from pi camera
105     print("[CAMERA] START...")
106     vd = VideoStream(src=0).start()
107     sleep(1)
108
109     while True:
110
111         # FPS start time
112         fps_start = time()
113
114         # Read a frame
115         frame = vd.read()
116
117         # Resize the frame
118         frame = resize_frame(frame, 60)
119
120         # Convert the frame to Gray scale
121         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
122
123         # Convert the frame dimensions to a blob and 200x200 dim
124         (height, width) = frame.shape[:2]
125         blob = cv2.dnn.blobFromImage
            (cv2.resize(frame, (200, 200)), 1.0,
            (224, 224), (104.0, 177.0, 123.0))
126
127         # Pass the blob through the deep neural
            # net(ResNet) and make the detections
128         net.setInput(blob)
129         detections = net.forward()
```

Γραμμή 106: Ξεκινάμε την ροή βίντεο από την $1^{η}$ κάμερα (src=0) που βρίσκεται συνδεδεμένη.

Γραμμή 115: Διαβάζουμε ένα πλαίσιο από την ροή.

Γραμμή 118: Αλλάζουμε το μέγεθος του πλαισίου κατά 60%.

Γραμμή 121: Μετατρέπουμε το πλαίσιο σε γκρι ώστε να μπορέσουμε να το επεξεργαστούμε.

Γραμμή 124-125: Αρχικά κρατάμε τις διαστάσεις της εικόνας σε μια πλειάδα (tuple) και ύστερα καλούμε την μέθοδο **'blobFromImage'** στην οποία περνάμε 4 παραμέτρους. Σαν $1^{η}$ παράμετρο περνάμε το πλαίσιο αφού το έχουμε αλλάξει το μέγεθος του σε 200x200 pixels, η $2^{η}$ παράμετρος αφορά τον πολλαπλασιαστή για τις τιμές του πλαισίου και στην περίπτωση μας είναι ίσος με 1.0, η $3^{η}$ παράμετρος αφορά το χωρικό μέγεθος της εικόνας εξόδου και η $4^{η}$ παράμετρος αφορά τις 3 μέσες τιμές (με σειρά B,G,R) η οποίες θα αφαιρεθούν από το πλαίσιο. Το αντικείμενο που επιστρέφεται είναι ένα 4 διαστάσεων Mat αντικείμενο σε NCHW σειρά (batch_shape + [height, width, channels]). [41]

Γραμμή 128-129: Θέτουμε ως είσοδο στο νευρωνικό δίκτυο το αντικείμενο blob που δημιουργήσαμε προηγουμένως και έπειτα καλούμε την μέθοδο forward() ώστε να πραγματοποιήσουμε τις ανιχνεύσεις.

```
132         # Loop over the detections to extract specific confidence
133         for i in range(0, detections.shape[0]):
134             confidence = detections[0, 0, i, 2]
135 
136             # Check the confidence
137             if confidence < 0.3:
138                 continue
```

Γραμμή 133: Ξεκινάμε να τρέχουμε ένα βρόχο για τις τρέχον ανιχνεύσεις του πλαισίου.

Γραμμή 134-138: Κρατάμε σε μια μεταβλητή το βαθμό εμπιστοσύνης της πρόβλεψης (confidence), και εάν αυτός ο βαθμός είναι μικρότερος από 0.3 (κλίμακα 0.0 έως 1.0) τότε παραλείπουμε τον υπόλοιπο κώδικα μέσα στον βρόχο και τρέχουμε την επόμενη επανάληψη.

```
140             # Compute the coordinates
141             box = detections[0, 0, i, 3:7] * np.array
                ([width, height, width, height])
142             (x, y, w, h) = box.astype("int")
143 
144             #Finding the distance between the camera and face
145             Distance = Distance_finder(Focal_length, KNOWN_WIDTH,w-x)
```

Γραμμή 141-142: Υπολογίζουμε τις συντεταγμένες όπου πιθανόν βρίσκεται το πρόσωπο και τις κρατάμε στην μεταβλητή box, και έπειτα μετατρέποντας τις συντεταγμένες σε ακέραιους, τις κρατάμε σε μια πλειάδα (x,y,w,h), όπου x,y συντεταγμένες για την επάνω αριστερή γωνία του πλαισίου και w,h το πλάτος (width) και το ύψος (height).

Γραμμή 145: Χρησιμοποιώντας τις μεταβλητές που είδαμε παραπάνω και με βάση τις συντεταγμένες, βρίσκουμε την απόσταση του προσώπου του οδηγού από την κάμερα.

```
161             # Create a rectangle object
162             rect = dlib.rectangle(int(x), int(y), int(w), int(h))
```

Γραμμή 162: Από τις συντεταγμένες δημιουργούμε ένα τετράγωνο, ώστε να μπορέσουμε να το αναπαραστήσουμε γραφικά γύρο από το πρόσωπο παρακάτω.

```
175             # Determines the facial landmarks in frame and
                # convert the x,y coordinates to a NumPy array
177             shape = face_predictor(gray, rect)
178             shape = shape_to_numpy(shape)
```

Γραμμή 177: Καλούμε το αντικείμενο face_predictor το οποίο αναφέρεται στην κλάση shpe_predictor και με την σειρά του καλεί την built-in μέθοδο __call__, με $1^{η}$ παράμετρο το gray scale πλαίσιο που μετατρέψαμε παραπάνω και $2^{η}$ παράμετρο τις συντεταγμένες όπου βρίσκεται το πρόσωπο. Η μέθοδος ανιχνεύει τα 68 σημεία πάνω στο πρόσωπο όπως αυτά έχουν καθοριστεί. Τέλος μας επιστρέφει ένα αντικείμενο το οποίο περιέχει τις συντεταγμένες για τα 68 αυτά σημεία πάνω στο πλαίσιο.

Γραμμή 178: Μετατρέπουμε τις συντεταγμένες της πρόβλεψης σε NumPy πίνακα.

```
186             # Extract left and right eye coordinates
187             leftEye  = EyesDetector.extract_left_coordinates(shape)
188             rightEye = EyesDetector.extract_right_coordinates(shape)
189
190             # Compute the Eye Aspect Ratio(E.A.R)
191             leftEAR = EyesDetector.eye_aspect_ratio(leftEye)
192             rightEAR = EyesDetector.eye_aspect_ratio(rightEye)
193
194             # Average for both eyes
195             avgEar = (leftEAR + rightEAR) / 2.0
```

Γραμμή 187-188: Λαμβάνουμε τις συντεταγμένες των σημείων για το αριστερό και δεξί μάτι.

Γραμμή 191-192: Υπολογίζουμε την αναλογία της απόστασης για κάθε μάτι, χρησιμοποιώντας την συνάρτηση **'eye_aspect_ratio'** (όπως είδαμε στο κεφάλαιο 6.3 → EyesDetection.py → Συνάρτηση eye_aspect_ratio).

Γραμμή 195: Υπολογίζουμε την μέση τιμή των ματιών.

```
216             # Extract lips coordinates
217             Lips = LipsDetector.extract_lips_coordinates(shape)
218
219             # Compute the Lips Aspect Ratio(L.A.R)
220             LipsLAR = LipsDetector.lips_aspect_ratio(Lips)
221             #print(f"LAR: ", LipsLAR)
```

Γραμμή 217: Λαμβάνουμε τις συντεταγμένες των σημείων για τα χείλη.

Γραμμή 220: Υπολογίζουμε την αναλογία της απόστασης των χειλιών, χρησιμοποιώντας την συνάρτηση 'lips_aspect_ratio' (όπως είδαμε στο κεφάλαιο 6.3 → LipsDetection.py → Συνάρτηση lips_aspect_ratio).

```
235             # Check EYES for drowsiness detection
236             if WARNING_BOTH_HANDS == False and LipsDetector.FLAG == False:
                    EyesDetector.detect_eyes_drowsiness \
                    (frame, avgEar, EYE_THRESHOLD, MAX_FRAMES_EYE,
                     alarm, TotalDetections, FINAL_FLAG)
240
241             # Check YAWN drowsiness detection
242             if WARNING_BOTH_HANDS == False and EyesDetector.FLAG == False:
                    LipsDetector.detect_yawn \
                    (frame, LipsLAR, LIPS_THRESHOLD, MAX_FRAMES_YAWN,
                     alarm, TotalDetections, FINAL_FLAG)
```

Γραμμή 236: Εάν δεν εντοπίζουμε τα δύο χέρια ταυτόχρονα σε περίεργη θέση και δεν εντοπίζουμε χασμουρητό τότε καλούμε την συνάρτηση **'detect_eyes_drowsines'** (όπως είδαμε στο κεφάλαιο 6.3 → EyesDetection.py → Συνάρτηση detect_eyes_drowsines) ώστε να ελέγξουμε για κλειστά μάτια.

Γραμμή 242: Εάν δεν εντοπίζουμε τα δύο χέρια ταυτόχρονα σε περίεργη θέση και δεν εντοπίζουμε κλειστά μάτια τότε καλούμε την συνάρτηση **'detect_yawn'** (όπως είδαμε στο κεφάλαιο 6.3 → LipsDetection.py → Συνάρτηση detect_yawn) ώστε να ελέγξουμε για χασμουρητό.

```
252             # Detect and find the body points with mediapipe framework
253             img = ArmHeadDetector.findPose(frame, False)
254             lmList = ArmHeadDetector.getPosition(img)
255             if len(lmList) != 0:
```

Γραμμή 252: Βρίσκουμε τα σημεία του σώματος μέσω της συνάρτησης **'findPose'** (όπως είδαμε στο κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση findPose).

Γραμμή 254-255: Επιστρέφουμε την λίστα με κάθε σημείο του σώματος μέσω της συνάρτησης **'getPosition'** (όπως είδαμε στο κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση getPosition) και ελέγχουμε εάν υπάρχουν στοιχεία στην λίστα ώστε να συνεχίσουμε.

```
257                 # Find the distance from right ear to right point of hand
258                 # Points:
259                 #   8  : Right Ear
260                 #   20 : Right Hand
261                 RightHEAR = ArmHeadDetector.hand_ear_aspect_ratio
                                (img, 20, 8, True)

265                 # Find the distance from left ear to left point of hand
266                 # Points:
267                 #   7  : Left Ear
```

```
268                 #   19 : Left Hand
269                 LeftHEAR = ArmHeadDetector.hand_ear_aspect_ratio
                              (img, 19, 7, True)
```

Γραμμή 261 & 269: Βρίσκουμε την απόσταση μεταξύ του αυτιού και του χεριού χρησιμοποιώντας την συνάρτηση **'hand_ear_aspect_ratio'** (όπως είδαμε στο κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση hand_ear_aspect_ratio) με παραμέτρους τα σημεία του δεξιού αυτιού (σημείο 8) και του δεξιού χεριού (σημείο 20) και αντίστοιχα τα σημεία του αριστερού αυτιού (σημείο 7) και αριστερού χεριού (σημείο 19).

```
273                 # Find the angle of right arm
274                 # Points:
275                 #   12 : Right shoulder
276                 #   14 : Right elbow
277                 #   16 : Right wrist
278                 RightARM = ArmHeadDetector.findAngle
                              (frame, 12, 14, 16, True)
280
281                 # Convert the number of angle into
                    # a new number system from 0 to 100
282                 # Right arm outside position
283                 RightARM_1 =
                    int(np.interp(RightARM,(250,360),(0,100)))
284
285
286                 # Find the angle of left arm
287                 # Points:
288                 #   11: left_shoulder
289                 #   13: left_elbow
290                 #   15: left_wrist
291                 LeftARM = ArmHeadDetector.findAngle
                             (frame, 11, 13, 15, True)
```

Γραμμή 278 & 290: Βρίσκουμε τις μοίρες της γωνίας που σχηματίζει το δεξί χέρι, μεταξύ των 3 παραμέτρων, που είναι ο δεξιός ώμος (σημείο 12), ο δεξιός αγκώνας (σημείο 14) και ο δεξιός καρπός (σημείο 16) χρησιμοποιώντας την συνάρτηση **'findAngle'** (όπως είδαμε στο κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση findAngle. Αντίστοιχα στην γραμμή 290, βρίσκουμε τις μοίρες της γωνίας το αριστερό χέρι, μεταξύ των 3 παραμέτρων, που είναι ο αριστερός ώμος (σημείο 11), ο αριστερός αγκώνας (σημείο 13) και ο αριστερός καρπός (σημείο 15).

Γραμμή 283: Μετατρέπουμε τον αριθμό των μοιρών που βρήκαμε για το δεξί χέρι στην γραμμή 283, σε μια κλίμακα από το 0 έως το 100, ώστε να μπορούμε να χρησιμοποιήσουμε τον αριθμό πιο κατανοητά παρακάτω.

```
299                 '''
300                 RIGHT HAND DETECTION
301                 '''
302                 if EyesDetector.FLAG == False
```

```
                    and LipsDetector.FLAG == False:
303
304                     # Checks if the right hand is in a good position
305                     RightHandDetector.detect_incorrect_right_arm \
                        (frame, [RightARM_1, RightARM],[30, [140, 180]],
                        [MAX_FRAMES_HEAR, 30], alarm, TotalDetections,
                         WARNING_BOTH_HANDS, FINAL_FLAG)
307
308                    # Checks if the right hand is close to the head
309                    # (Driver talks to phone or had strange bahaviour)
310                    if RightHandDetector.FLAG == False:
311                         RightPhoneDetector.detect_phone_hand \
                            (frame, RightHEAR, HEAR_THRESHOLD,
                            MAX_FRAMES_HEAR, alarm, TotalDetections,
                            WARNING_BOTH_HANDS, FINAL_FLAG)
```

Γραμμή 305: Καλούμε την συνάρτηση **'detect_incorrect_right_arm'** με παραμέτρους που έχουμε δει προηγουμένως (κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση detect_incorrect_right_arm), ώστε να δούμε εάν το δεξί χέρι βρίσκεται σε μη κανονική θέση.

Γραμμή 311: Καλούμε την συνάρτηση **'detect_phone_hand'** (κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση detect_phone_hand), ώστε να εντοπίσουμε πιθανόν χρήση κινητού τηλεφώνου στο δεξί χέρι ή εντοπισμό δεξιού χεριού σε μη κανονική θέση.

```
315                 '''
316                 LEFT HAND DETECTION
317                 '''
318                 if EyesDetector.FLAG == False
                    and LipsDetector.FLAG == False:
319
320                     # Checks if the left hand is in a good position
321                     LeftHandDetector.detect_incorrect_left_arm \
                        (frame, LeftARM, [[10, 70], [190, 360]],
                         30, alarm, TotalDetections,
                         WARNING_BOTH_HANDS, FINAL_FLAG)
323
324                     # Checks if the left hand is close to the head
325                     # (Driver talks to phone or had strange behavior)
326                     if LeftHandDetector.FLAG == False:
327                         LeftPhoneDetector.detect_phone_hand \
                            (frame, LeftHEAR, HEAR_THRESHOLD,
                            MAX_FRAMES_HEAR, alarm, TotalDetections,
                            WARNING_BOTH_HANDS, FINAL_FLAG)
```

Γραμμή 321: Καλούμε την συνάρτηση **'detect_incorrect_left_arm'** με παραμέτρους που έχουμε δει προηγουμένως (κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση detect_incorrect_left_arm), ώστε να δούμε εάν το αριστερό χέρι βρίσκεται σε μη κανονική θέση.

Γραμμή 327: : Καλούμε την συνάρτηση **'detect_phone_hand'** (κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση detect_phone_hand), ώστε να εντοπίσουμε πιθανόν χρήση κινητού τηλεφώνου στο αριστερό χέρι ή εντοπισμό αριστερού χεριού σε μη κανονική θέση.

```
364                     # Checks if the driver's head tils to the right side
365                     TiltedHeadRight.detect_phone_hand \
                        (frame, rightTAR, TAR_THRESHOLD, MAX_FRAMES_TAR,
                         alarm, TotalDetections,  WARNING_BOTH_HANDS,
                         FINAL_FLAG)
…
376                     # Check if the driver's head tils to the left side
377                     TiltedHeadLeft.detect_phone_hand \
                        (frame, leftTAR, TAR_THRESHOLD, MAX_FRAMES_TAR,
                         alarm, TotalDetections, WARNING_BOTH_HANDS,
                         FINAL_FLAG)
```

Γραμμή 365 & 377: Καλούμε την συνάρτηση **'detect_phone_hand'** (κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση detect_phone_hand) με διαφορετικές παραμέτρους, έτσι ώστε να εντοπίσουμε πιθανόν κλίση του κεφαλιού προς την δεξιά (Γραμμή 365) ή αριστερή (Γραμμή 377) πλευρά του ώμου.

```
381             '''
                If the BOTH HANDS have been detected as strange behaviour
                then we call a specific function for this purpose
                '''
385                  if ((RightHandDetector.BOTH_HANDS == True
                     and LeftHandDetector.BOTH_HANDS == True) or
                        (RightHandDetector.BOTH_HANDS == True
                     and LeftPhoneDetector.BOTH_HANDS == True) or
                        (RightPhoneDetector.BOTH_HANDS == True
                     and LeftHandDetector.BOTH_HANDS == True)  or
                        (RightPhoneDetector.BOTH_HANDS == True
                     and LeftPhoneDetector.BOTH_HANDS == True)):
389                    WARNING_BOTH_HANDS = True
390                    ArmHeadDetector.both_hands_detect
                        (frame, alarm, TotalDetections, FINAL_FLAG)
```

Γραμμή 385: Βλέπουμε εάν υπάρχει εντοπισμός ταυτόχρονα και για τα δύο χέρια.

Γραμμή 390: Σε αυτήν την περίπτωση καλούμε την συνάρτηση **'both_hands_detect'** (κεφάλαιο 6.3 → ArmHeadDetection.py → Συνάρτηση both_hands_detect).

7.5 Σύγκριση μοντέλων εντοπισμού προσώπου

Ο πρώτος στόχος που είχαμε κατά την διάρκεια υλοποίησης του έργου ήταν αρχικά να μπορέσουμε να εντοπίζουμε το πρόσωπο του οδηγού όσο το δυνατόν πιο καλύτερα γίνεται, εννοώντας να μπορούμε να εντοπίζουμε το πρόσωπο συνεχώς με τα λιγότερα δυνατά σφάλματα εντοπισμού. Βέβαια ένα ακόμα στόχος που θέσαμε ήταν να χρησιμοποιήσουμε ένα μοντέλο το οποίο θα μπορούσε να τρέχει σε πραγματικό χρόνο πάνω στο Raspberry Pi, καθώς όπως γνωρίζουμε το υλικό του Raspberry Pi μας περιορίζει, ώστε να τρέξουμε κάποιο μοντέλο το οποίο απαιτεί μεγάλη επεξεργαστική ισχύ. Με αυτόν τον τρόπο καταλήξαμε στα μοντέλα Haar Cascades και Caffe model.

Αρχικά και τα 2 μοντέλα χρησιμοποιούνται για τον εντοπισμό προσώπου. Η διαφορά τους είναι ότι το μοντέλο Haar Cascade είναι ουσιαστικά ένα αλγόριθμος, όπου εφαρμόζει μια συγκεκριμένη τεχνική ώστε να εντοπίζει ένα πρόσωπο, ενώ το Caffe model είναι ένα μοντέλο που χρησιμοποιείται με βάση το αλγόριθμο SSD (Single Shot Detector), ο οποίος βασίζεται στην αρχιτεκτονική δικτύου ResNet και χρησιμοποιεί ένα βαθύ νευρωνικό δίκτυο για τον εντοπισμό των αντικειμένων του.

Το πρώτο μοντέλο που είδαμε αρχικά είναι το μοντέλο Haar Cascade, όπου μπορούμε να το βρούμε στο επίσημο αποθετήριο(repository) του OpenCV, το οποίο βρίσκεται στο GitHub(https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml). Το μοντέλο βρίσκεται σε μορφή αρχείου XML (Extensible Markup Language) και μπορούμε να το κατεβάσουμε και να το χρησιμοποιήσουμε.

Το δεύτερο μοντέλο που χρησιμοποιήθηκε είναι το Caffe model, το οποίο μπορούμε να βρούμε στο GitHub (https://github.com/vinuvish/Face-detection-with-OpenCV-and-deep-learning/tree/master/models). Στο αποθετήριο θα βρούμε 2 αρχεία, το αρχείο .prototxt το οποίο ορίζει την αρχιτεκτονική του μοντέλου, και το αρχείο .caffemodel το οποίο αφορά τα βάρη που θα χρησιμοποιήσουν τα επίπεδα (layers) του μοντέλου.

Χρησιμοποιώντας το μοντέλο Haar Cascade αρχικά μπορούμε να επιτύχουμε μέχρι και 15 πλαίσια ανά δευτερόλεπτο (FPS), εάν θέλουμε να χρησιμοποιήσουμε μόνο αυτό το μοντέλο. Στην περίπτωση μας στην τελική έκδοση του προγράμματος μας, υπήρχε μεγαλύτερη πολυπλοκότητα στον κώδικα μας, καθώς χρησιμοποιήσαμε ένα ακόμη μοντέλο μηχανικής μάθησης (MediaPipe Pose). Αφού υπολογίσαμε των αριθμό

πλαισίων ανά δευτερόλεπτο, εν τέλει είχαμε ένα αποτέλεσμα των 3 πλαισίων ανά δευτερόλεπτο (FPS), αυτό σημαίνει ότι το πρόγραμμα μας ήταν σε θέση να επεξεργάζεται και να εμφανίζει ∼3 πλαίσια ανά δευτερόλεπτο στην οθόνης μας. Αυτό σχετικά μας έδινε ένα ταχύτερο αποτέλεσμα σε σχέση με το μοντέλο Caffe model το οποίο μας έδινε μόλις ∼2 πλαίσια ανά δευτερόλεπτο (FPS). Η κύρια διαφορά τους όμως είναι ότι το Caffe model είχε πολύ μεγαλύτερη ακρίβεια στο εντοπισμού του προσώπου, έχοντας σχεδόν μηδενικό αριθμό σφάλματων στο να εντοπίσει κάτι το οποίο δεν ήταν πρόσωπο. Σε αντίθεση το μοντέλο Haar Cascade είχε πολλά σφάλματα στο να εντοπίζει αντικείμενα τα οποία δεν αφορούσαν πρόσωπο, με αποτέλεσμα ορισμένες φορές, να μην εντοπίζει το πρόσωπο του οδηγού και να δίνει εσφαλμένα αποτελέσματα. Επομένως για τον εντοπισμού του προσώπου καταλήξαμε να χρησιμοποιήσουμε το Caffe model, το οποίο θα μας έδινε μεγαλύτερη ακρίβεια με μόνο μειονέκτημα το χρόνο.

7.6 Εφαρμογή του έργου μας

Σε αυτήν την υποενότητα, θα τοποθετήσουμε το σύστημα μας σε ένα αυτοκίνητο, ώστε να το εφαρμόσουμε σε πραγματικές συνθήκες λειτουργίας, θα δούμε διαφορετικά παραδείγματα λειτουργίας του, καθώς και τον τρόπο με τον οποίο ανταποκρίνεται σε όλες τις περιπτώσεις εντοπισμού.

7.6.1 Τοποθέτηση του συστήματος σε όχημα

Μεγάλο ρόλο παίζει η τοποθέτηση του συστήματος, και πιο συγκεκριμένα της κάμερας που θα παρακολουθεί τον οδηγό, καθώς η ‘’λάθος’’ τοποθέτηση του μπορεί να οδηγήσει σε εσφαλμένους εντοπισμούς. Το σημείο τοποθέτησης του συστήματος σε κάθε αυτοκίνητο πιθανόν να διαφέρει. Επίσης, να αναφέρουμε πως το σύστημα έχει δημιουργηθεί με βάση τα αυτοκίνητα τα οποία έχουν το τιμόνι τους αριστερά, η τοποθέτηση λοιπόν σε ένα αυτοκίνητο με το τιμόνι δεξιά θα απαιτούσε την αλλαγή ορισμένων παραμέτρων.

Στην περίπτωσή μας το σύστημα μας τοποθετήθηκε στο αυτοκίνητο χρησιμοποιώντας τις παρακάτω αποστάσεις:

1. 45 εκατοστά (cm) δεξιά από το κέντρο του τιμονιού
2. 15 εκατοστά (cm) κάτω από το ύψος των ματιών του οδηγού
3. 95 εκατοστά (cm) απόσταση από τον οδηγό

7.6.2 Τροφοδοσία του συστήματος

Στην περίπτωση μας χρησιμοποιήσαμε παροχή ρεύματος ώστε να πραγματοποιηθούν οι δοκιμές και να καταγραφούν screenshots και λειτουργίας του συστήματος.

Για να μπορέσουμε να χρησιμοποιούμε το σύστημα μας, θα πρέπει το Raspberry Pi να τροφοδοτείται επαρκώς (5 Volt, 3 Ampere, USB Type C) κατά την διάρκεια της χρήσης του. Να υπενθυμίσουμε πως οι απαιτήσεις για την τροφοδοσία ενός Raspberry Pi 4 είναι:

- Η ελάχιστη ισχύς εξόδου θα πρέπει να είναι 5V
- Τουλάχιστον 3 Ampere
- Θα πρέπει να χρησιμοποιείται καλώδιο με σύνδεση USB τύπου C (με E-mark-chip)

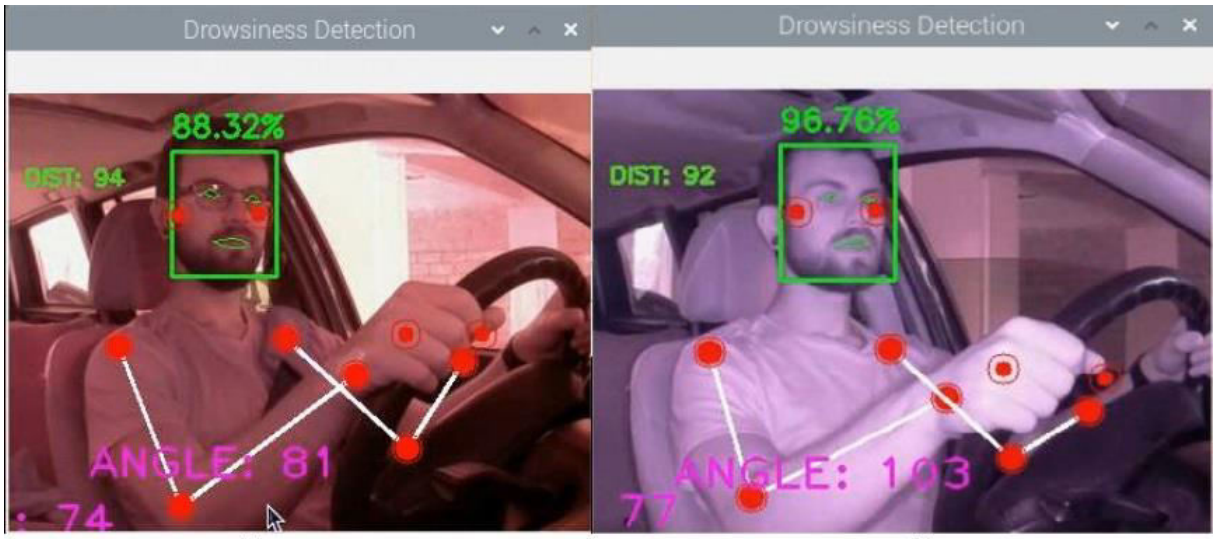
Υπάρχουν αρκετοί τρόποι ώστε να τροφοδοτήσουμε το Raspberry Pi 4, κάποιοι από τους οποίους είναι:

1. Τροφοδοσία του Raspberry Pi 4, χρησιμοποιώντας Power Bank, τα οποία μπορούν να δώσουν τουλάχιστον 10 Watt ισχύ και με χωρητικότητα τουλάχιστον 20000 mAh (~12 ώρες δυνατότητας τροφοδοσίας).
2. Τροφοδοσία του Raspberry Pi 4, χρησιμοποιώντας ένα τροφοδοτικό αντάπτορα αυτοκινήτου (car adapter power supply) , ο οποίος φυσικά θα έχει τις απαιτήσεις που περιγράψαμε παραπάνω (>= 5 Volt, >= 3 Ampere, USB Type C with E-mark-chip)

7.6.3 Λειτουργία του συστήματος

Παρακάτω θα δούμε ένα εύρος παραδειγμάτων με εικόνες και θα εξηγήσουμε πως ανταποκρίνεται το σύστημα μας, σε κάθε περίπτωση εντοπισμού. Η διαδικασία με την οποία λειτουργεί το σύστημα, έχει περιγραφή με το UML διάγραμμα δραστηριότητας (Ενότητα 6.3).

Αρχικά, στην Εικόνα 7.6 (a) & (b) βλέπουμε μια αρχική θέση του οδηγού, η οποία υποθέτουμε πως είναι η σωστή θέση που πρέπει να έχει ένας οδηγός μέσα στο αυτοκίνητο. Επίσης, βλέπουμε και τα χαρακτηριστικά του οδηγού που έχουμε εντοπίσει, και τα αναπαριστούμε γραφικά στο πλαίσιο. Με βάση αυτά τα χαρακτηριστικά θα κρίνουμε την συμπεριφορά του οδηγού κατά την οδήγηση. Ακόμα όπως φαίνεται στην Εικόνα 7.6 (a), ο οδηγός φοράει γυαλιά οράσεως, το οποίο δεν επηρεάζει το σύστημα ως προς τον εντοπισμό των ματιών και στην Εικόνα 7.6 (b), ο εντοπισμός πραγματοποιείται κατά την διάρκεια της νύχτας. Στο πράσινο τετράγωνο βλέπουμε τον εντοπισμό του προσώπου του οδηγού με τα χαρακτηριστικά του, και ακριβώς από επάνω το ποσοστό ακρίβειας του εντοπισμού του (%). Ακριβώς αριστερά αναπαριστούμε την απόσταση του οδηγού από την κάμερα (DIST). Παρακάτω βλέπουμε την γραφική αναπαράσταση των σημείων των χεριών που έχουμε εντοπίσει, καθώς και τις γωνίες κάθε χεριού (ANGLE).



(a) (b)

Εικόνα 7.6 Κανονική θέση του οδηγού στο αυτοκίνητο (a) κατά την διάρκεια της μέρας (b) κατά την διάρκεια της νύχτας

Παρακάτω θα δούμε εικόνες, στις οποίες έχουμε εντοπίσει τον οδηγό με συμπεριφορά υπνηλίας ή κόπωσης. Κάθε φορά που πραγματοποιείται ο εντοπισμός, αυξάνουμε και ένα μετρητή προειδοποιήσεων, ο οποίος όπως έχουμε δει στο διάγραμμα 6.3, εκτελεί μια συγκεκριμένη διαδικασία. Στην Εικόνα 7.7 εντοπίζουμε τον οδηγό καθώς χασμουριέται. Ο εντοπισμός του χασμουρητού απαιτεί τουλάχιστον ~ 1,5 δευτερόλεπτα για τον εντοπισμό του.

Εικόνα 7.7 Εντοπισμός χασμουρητού οδηγού

Στην Εικόνα 7.8 εντοπίζουμε τον οδηγό να έχει κλειστά μάτια. Εντοπίζουμε κλειστά μάτια εφόσον παραμένουν κλειστά για τουλάχιστον ~2,5 δευτερόλεπτα.

Εικόνα 7.8 Εντοπισμός κλειστών ματιών οδηγού (a) χωρίς γυαλιά (b) με γυαλιά οράσεως

Παρακάτω θα δούμε εικόνες στις οποίες έχουμε εντοπίσει τον οδηγό με κάποια μη κανονική συμπεριφορά. Στην Εικόνα 7.9 εντοπίζουμε τον οδηγό να μιλάει στο κινητό του, και στις δύο περιπτώσεις. Στην Εικόνα 7.9 (a) βλέπουμε τον οδηγό να κρατάει το κινητό στο δεξί χέρι και στην 7.9 (b) στον αριστερό χέρι αντίστοιχα. Για να εντοπίσουμε χρήση κινητού θα πρέπει να βλέπουμε την κίνηση αυτή του οδηγού για τουλάχιστον ~15 δευτερόλεπτα.

Εικόνα 7.9 Εντοπισμός χρήσης κινητού (a) στο δεξί χέρι & (b) στο αριστερό χέρι

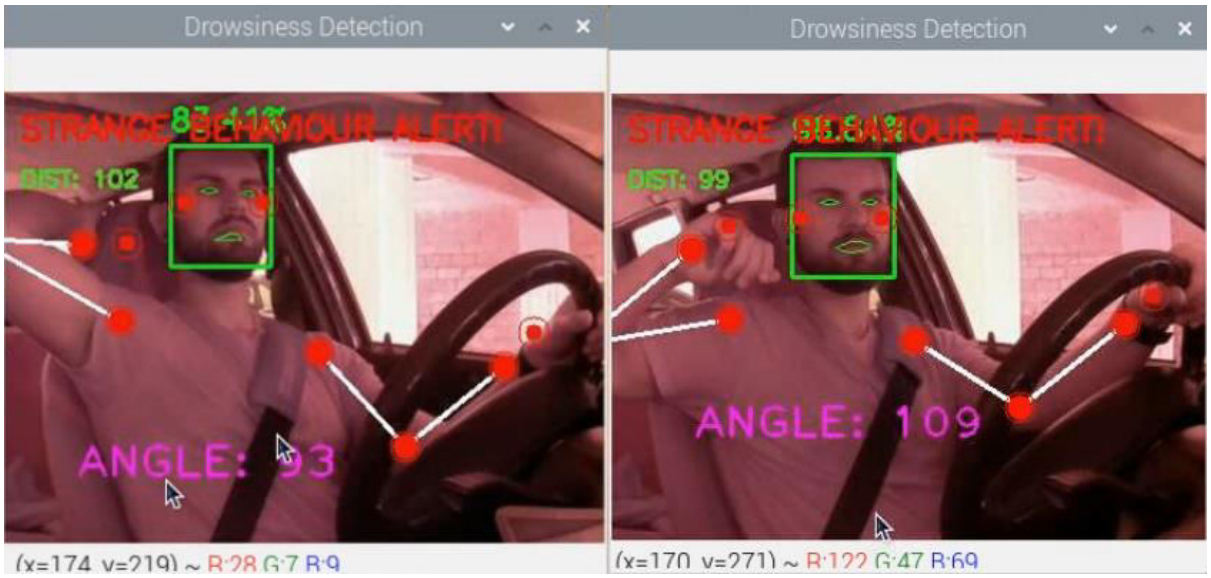
Στην Εικόνα 7.10 εντοπίζουμε το κεφάλι του οδηγού να έχει μια κλίση προς τα αριστερά ή δεξιά αντίστοιχα. Αυτό μας δείχνει μια περίεργη συμπεριφορά του οδηγού

και θεωρούμε πως ο οδηγός είναι κουρασμένος. Σε μια τέτοια περίπτωση, ο οδηγός πιθανόν να έχει κλειστά τα μάτια του και να είναι αυτό το χαρακτηριστικό το οποίο θα εντοπιστεί πρώτο. Αλλά στην περίπτωση μας εντοπίζεται από την κλίση που έχει το κεφάλι του.

Εικόνα 7.10 Εντοπισμός περίεργης συμπεριφοράς καθώς (a) το κεφάλι έχει κλίση προς τον δεξιό ώμο & (b) κεφάλι έχει κλίση προς τον αριστερό ώμο

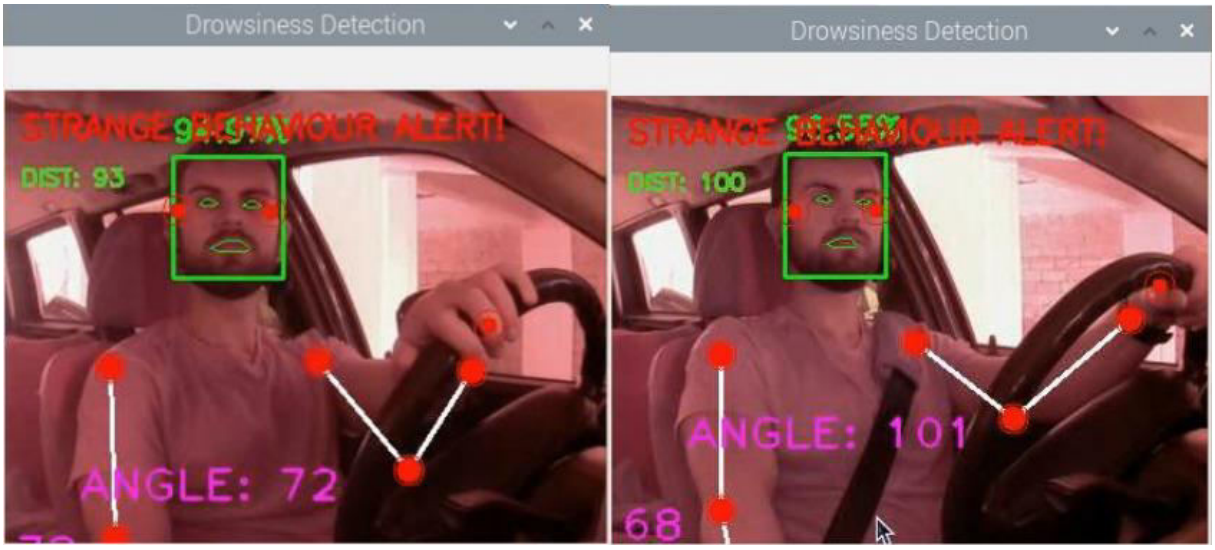
Στην Εικόνα 7.10 (a) βλέπουμε το κεφάλι του οδηγού να έχει μια κλίση προς τα δεξιά και στην Εικόνα 7.10 (b) βλέπουμε το κεφάλι του οδηγού να έχει μια κλίση προς τα αριστερά. Αυτό που κοιτάμε εμείς πραγματικά, ώστε να πραγματοποιήσουμε τον εντοπισμό είναι η απόσταση που έχει το σημείο του αυτιού του οδηγού, από το σημείο του αντίστοιχου ώμου του. Ο εντοπισμός πραγματοποιείται εφόσον περάσουν τουλάχιστον ~5 δευτερόλεπτα.

Στην Εικόνα 7.11 και στις δύο περιπτώσεις εντοπίζουμε το χέρι του οδηγού να βρίσκεται εκτός του τιμονιού και σε περίεργη θέση.



Εικόνα 7.11 Εντοπισμός περίεργης συμπεριφοράς καθώς το χέρι και στις δύο περιπτώσεις βρίσκεται σε περίεργη θέση

Στην Εικόνα 7.12 βλέπουμε μια περίπτωση εντοπισμού που αφορά το δεξί χέρι το οποίο βρίσκεται για αρκετή ώρα εκτός τιμονιού και στην περίπτωση μας στον λεβιέ ταχυτήτων. Προφανώς δεν εντοπίζουμε την κίνηση του οδηγού, για το μικρό χρονικό διάστημα που αλλάζει ταχύτητα. Αυτό που συμβαίνει όμως πολλές φορές, είναι να έχουμε το χέρι στο λεβιέ ταχυτήτων ή οπουδήποτε αλλού (για μεγάλο χρονικό διάστημα) σε μια σταθερή πορεία, αλλά αποτελεί λάθος, καθώς θα πρέπει να έχουμε τα χέρια στην σωστή θέση πάνω στο τιμόνι.



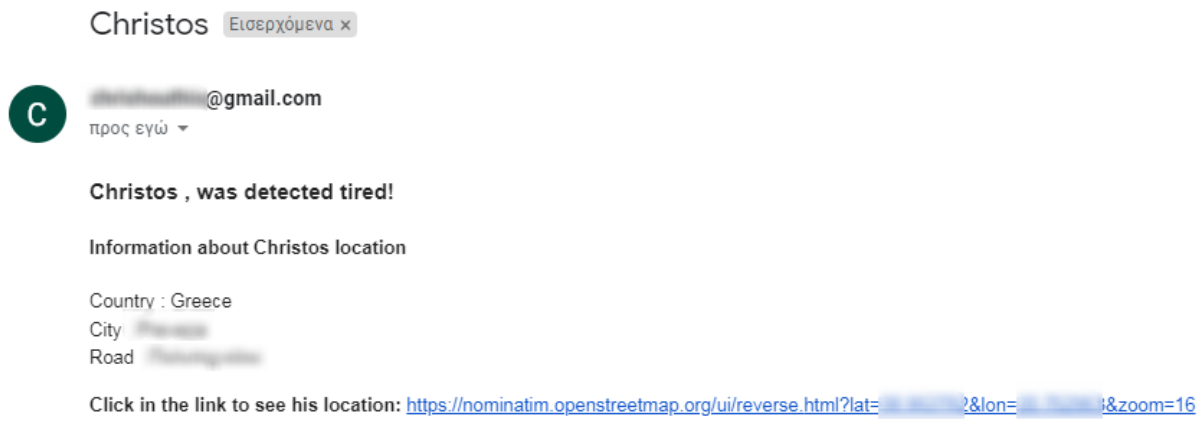
Εικόνα 7.12 Εντοπισμός περίεργης συμπεριφοράς και στις δύο περιπτώσεις καθώς το χέρι βρίσκεται στο λεβιέ ταχυτήτων ή αλλού σε μη κανονική θέση για μεγάλο χρονικό διάστημα

Στην Εικόνα 7.13 και στις δύο περιπτώσεις εντοπίζουμε τα χέρια του οδηγού να βρίσκονται εκτός του τιμονιού και σε περίεργη θέση.

Εικόνα 7.12 Εντοπισμός περίεργης συμπεριφοράς και στις δύο περιπτώσεις, καθώς τα χέρια βρίσκονται σε περίεργη θέση για μεγάλο χρονικό διάστημα (Η λήψη των εικόνων έχει πραγματοποιηθεί σε σκοτάδι)

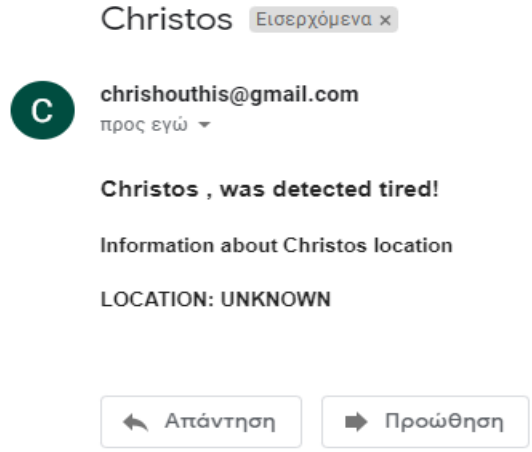
Όπως είδαμε και στο UML διάγραμμα δραστηριότητας (Ενότητα 6.3), το πράσινο LED βρίσκεται ενεργοποιημένο από την αρχή που τρέχουμε το πρόγραμμα και καθ' όλη την διάρκεια, εφόσον δεν υπάρχει κάποιος εντοπισμός. Εάν ο μετρητής είναι μικρότερος του 10 (μετρητής προειδοποιήσεων < 10), και πραγματοποιείται κάποιος εντοπισμός, τότε ενεργοποιείται στιγμιαία το κίτρινο LED. Έπειτα στην περίπτωση που ο μετρητής γίνει ίσος με 10 (μετρητής προειδοποιήσεων == 10), τότε ενεργοποιείται το κόκκινο LED. Αφού έχει ενεργοποιηθεί το κόκκινο LED, τρέχει ένα νήμα το οποίο είναι υπεύθυνο για την αποστολή του ηλεκτρονικού μηνύματος. Το νήμα αυτό αρχικά κοιτάζει ένα υπάρχει πρόσβαση στο διαδίκτυο (π.χ. μέσω Wi-Fi) ώστε να μπορέσει να συνεχίσει, αλλιώς στην περίπτωση που δεν υπάρχει πρόσβαση στο διαδίκτυο τότε ενεργοποιείται το A9G GSM/GPRS Module, ώστε να έχουμε πρόσβαση στο διαδίκτυο με χρήση δεδομένων κινητής τηλεφωνίας (Ενότητα 5.6). Έπειτα εφόσον υπάρχει πρόσβαση στο διαδίκτυο, ελέγχει εάν το GPS είναι ήδη ενεργοποιημένο, εφόσον το GPS είναι ενεργοποιημένο, με κατάλληλες εντολές επιστρέφονται οι συντεταγμένες (γεωμετρικό πλάτος και μήκος), όπου και χρησιμοποιούνται ώστε να διαμορφώσουν το μήνυμα που θα σταλεί. Τέλος το μήνυμα αποστέλλετε ηλεκτρονικά στο email του παραλήπτη που

έχουμε καθορίσει. Η μορφή του μηνύματος που λαμβάνει ο παραλήπτης, εφόσον οι συνταραγμένες έχουν εντοπισθεί φαίνεται στην Εικόνα 7.13.



Εικόνα 7.13 Μορφή μηνύματος που λαμβάνει ο παραλήπτης στην περίπτωση των 10 προειδοποιήσεων και εντοπισμού των συντεταγμένων

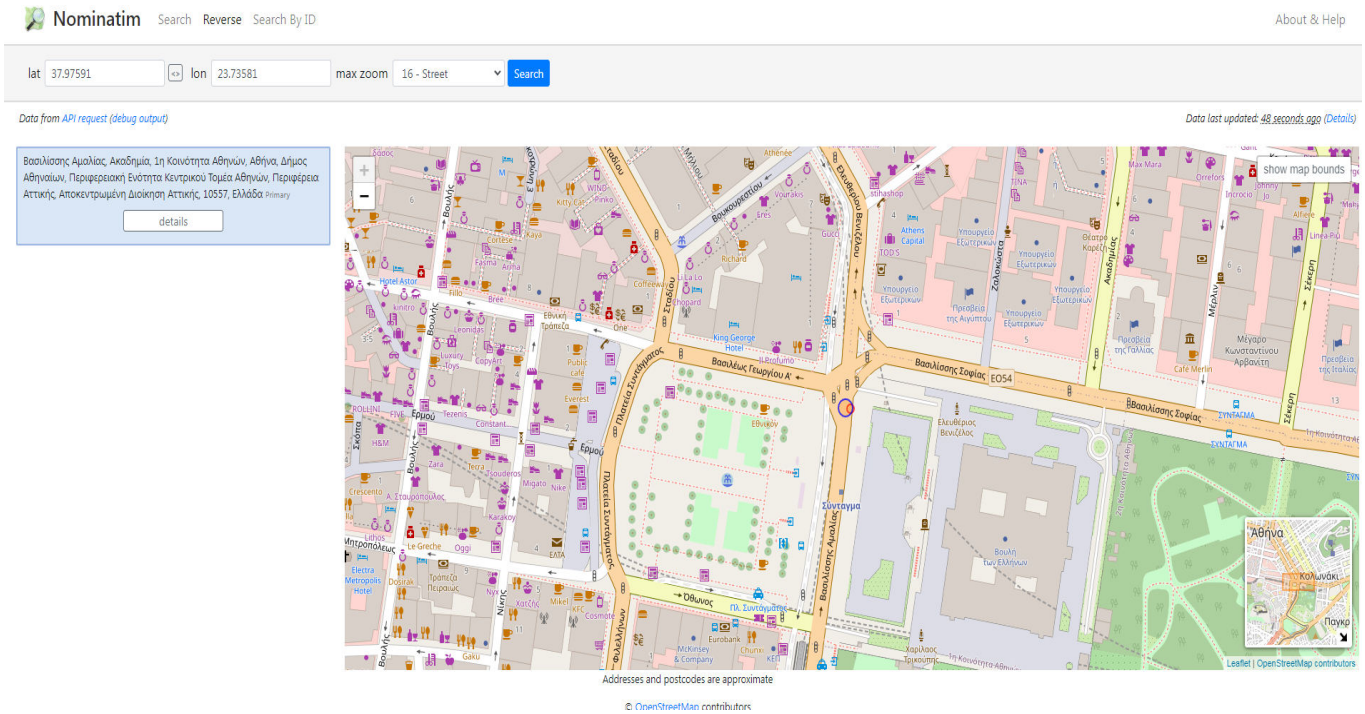
Σε άλλη περίπτωση, εφόσον οι συντεταγμένες δεν εντοπισθούν τότε ο παραλήπτης, λαμβάνει την μορφή του μηνύματος που φαίνεται στην Εικόνα 7.14.



Εικόνα 7.14 Μορφή μηνύματος που λαμβάνει ο παραλήπτης στην περίπτωση των 10 προειδοποιήσεων και ΜΗ εντοπισμού των συντεταγμένων

Όπως βλέπουμε στην Εικόνα 7.13 ο παραλήπτης λαμβάνει κάποιες πληροφορίες, πάντα με βάση τις διαθέσιμες που υπάρχουν και στο τέλος υπάρχει ένα κατάλληλα διαμορφωμένο link, το οποίο ο παραλήπτης μπορεί να ανοίξει, ώστε να δει την ακριβή τοποθεσία του οδηγού σε μορφή χαρτών μέσω του Nominatim. Επιπλέον, το link χρησιμοποιείται, ώστε ο παραλήπτης να μπορέσει να δει περισσότερες πληροφορίες. Παραδείγματος χάρη στην Εικόνα 7.15 βλέπουμε την μορφή την οποία θα αντικρίσει ο

παραλήπτης εφόσον ανοίξει το link (η τοποθεσία είναι τυχαία και αποτελεί παράδειγμα).



Εικόνα 7.15 Τοποθεσία χάρτη Nominatim μέσω των συντεταγμένων

7. Συμπεράσματα – Μελλοντικές Προτάσεις

Όπως είδαμε και αναφέραμε, πειραματιστήκαμε με διαφορετικά μοντέλα και μεθόδους ανίχνευσης προσώπου, εντοπισμού των χαρακτηριστικών προσώπου και χαρακτηριστικών σώματος, ώστε να δημιουργήσουμε ένα σύστημα το οποίο θα μπορούσε με όσο το δυνατόν μεγαλύτερη ακρίβεια, να εντοπίσει τον οδηγό και την συμπεριφορά του κατά της διάρκεια της οδήγησης.

Ώστοσο, το σύστημα αυτό μπορεί να βελτιωθεί στην πορεία, εν μέσω καινούριων μεθόδων εντοπισμού, και να τροποποιηθεί κατάλληλα για την χρήση της μικρότερης δυνατής υπολογιστικής ισχύς, όσο και για την δυνατόν καλύτερη προσέγγιση του. Παρακάτω θα δούμε κάποια στοιχεία που μπορούν να προστεθούν μελλοντικά, ώστε βελτιώσουν την απόδοση και την ακρίβεια του συστήματος.

Καθώς η κάμερα, βρίσκονταν σε διαφορετική κλίση (προς τα δεξιά), από αυτήν που συνήθως βλέπει ο οδηγός (ευθεία), παρατηρήσαμε πως υπήρχαν φορές που ο εντοπισμός των κλειστών ματιών, δεν πραγματοποιήθηκε, καθώς υπήρχε αυτή η γωνία μεταξύ της κάμερας και του προσώπου του οδηγού. Μια πιθανή λύση σε αυτό το πρόβλημα είναι, η τοποθέτηση μιας δεύτερης κάμερας, πάνω στο ταμπλό πίσω ακριβώς από το τιμόνι ή σε εναλλακτική θέση αντίστοιχα, ώστε τα χαρακτηριστικά του προσώπου του οδηγού να φαίνονται πιο κοντά, προσφέροντας ακριβέστερα αποτελέσματα.

Όπως είδαμε η υπολογιστική ισχύς παίζει σημαντικό ρόλο, ώστε να μπορούμε να τρέχουμε διαφορετικά μοντέλα, αναγνωρίζοντας και παρακολουθώντας τα χαρακτηριστικά του ατόμου, με όσο το δυνατόν μεγαλύτερη ταχύτητα. Σε ένα σύστημα όμως, όπως το Raspberry Pi δεν έχουμε πάντα την δυνατότητα να τρέχουμε μοντέλα και αλγορίθμους τα οποία απαιτούν μεγάλη υπολογιστική ισχύ. Ένας τρόπος για να αυξήσουμε λοιπόν την ταχύτητα του συστήματος μας, ώστε να μπορεί το σύστημα μας να επεξεργάζεται περισσότερα πλαίσια την φορά, είναι να χρησιμοποιήσουμε το λογισμικό OpenVINO (λογισμικό για την βελτίωσης απόδοσης αλγορίθμων βαθιάς μηχανικής μάθησης κτλ.) σε συνδυασμό με το Movidius NCS (USB Stick το οποίο επιτυγχάνει την ανάλυση αποτελεσμάτων σε συστήματα που τρέχουν αλγορίθμους βαθιάς μηχανικής μάθησης κτλ.). Επίσης, μπορούμε να αυξήσουμε των αριθμό των πλαισίων, χρησιμοποιώντας ένα ξεχωριστό νήμα για την διαδικασία του διαβάσματος

και της αποκωδικοποίησης του κάθε πλαισίου, σε συνδυασμό με την χρήση δομής δεδομένων ουράς.

Η Χρήση του A9G GSM/GPRS+GPS Module θα μπορούσε ίσως να αντικατασταθεί από δύο ξεχωριστά Module (GPS & GMS/GPRS), ώστε να μπορούν να λειτουργούν και τρέχουν ταυτόχρονα, καθώς στην περίπτωση μας έπρεπε να υπάρχει μια διαδικασία ώστε να ανοίγουμε/κλείνουμε την μια από της δύο λειτουργίες, πράγμα χρονοβόρο και μη αποδοτικό πολλές φορές. Ακόμα θα μπορούσε να χρησιμοποιηθεί κάρτα SIM παγκόσμιας εμβέλειας, ώστε ο χρήστης του συστήματος να έχει πρόσβαση σε δεδομένα, μέσω του συστήματος κινητής τηλεφωνίας σε παγκόσμια εμβέλεια.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] S. R. &. P. Norvig, Τεχνητή Νοημοσύνη “Μια σύγχρονη προσέγγιση”, Γ. Ρεφανίδη, Ed., Κλειδάριθμος, 2005.

[2] Ευρωπαικό Κοινοβούλιο, "Ευρωπαικό Κοινοβούλιο," 26 March 2021. [Online]. Available: https://www.europarl.europa.eu/news/el/headlines/society/20200827STO85804/ti-einai-i-techniti-noimosuni-kai-pos-chrisimopoieitai.

[3] Gravitonio, "Gravitonio," 6 January 2013. [Online]. Available: https://gravitonio.blogspot.com/2013/01/alan-turing.html.

[4] javaTpoint, "javaTpoint," [Online]. Available: https://www.javatpoint.com/turing-test-in-ai.

[5] Βικιπαίδεια, "Μηχανική όραση," 6 February 2020. [Online]. Available: https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CF%8C%CF%81%CE%B1%CF%83%CE%B7.

[6] analyticSteps, "analyticSteps," 18 March 2021. [Online]. Available: https://www.analyticssteps.com/blogs/8-popular-computer-vision-applications.

[7] [Online]. Available: https://repository.kallipos.gr/bitstream/11419/3382/1/02_chapter_04.pdf.

[8] Kamil Krzyk, 25 July 2018. [Online]. Available: https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d.

[9] A. Mansourian, A. Ghanizadeh and G. B., "Modeling of Resilient Modulus of Asphalt Concrete Containing Reclaimed Asphalt Pavement using Feed-Forward and Generalized Regression Neural Networks," p. 9, January 2018.

[10] Kumar,A.; Kaur,A.; Kumar,M.;, "Face detection techniques: a review," 4 August 2018.

[11] OpenCV, "Cascade Classifier," [Online]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[12] Adrian Rosebrock, "Face detection tips, suggestions, and best practices," 26 April 2021. [Online]. Available: https://www.pyimagesearch.com/2021/04/26/face-detection-tips-suggestions-and-best-practices/.

[13] Wikipedia, "Caffe (software)," 4 October 2021. [Online]. Available: https://en.wikipedia.org/wiki/Caffe_(software).

[14] Jonathan Hui, "SSD object detection: Single Shot MultiBox Detector for real-time processing," 14 March 2018. [Online]. Available: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.

[15] Chavdar Alexandrov, "Radar Imaging and Radar Image Processing," p. 4, August 2013.

[16] Liu,W.; Anguelov,D.;, Erhan,D.; Szegedy,C.; Reed,S.; Fu,C.; Berg,A.C.;, "SSD: Single Shot MultiBox Detector," p. 3, 29 December 2016.

[17] Dlib, 28 Μάρτιος 2021. [Online]. Available: http://dlib.net/. [Accessed 22 September 2021].

[18] Korshunov,P.;Marcel,S.;, "Speaker Inconsistency Detection in Tampered Video," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018.

[19] Kartynnik,Y.;Ablavatski,A.;Grishchenko,I.;Grundmann,M.;, "Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs," p. 1, 2019.

[20] Zheng,C.; Wu,W.; Yang,T.; Zhu,S.; Chen,C.; Liu,R.; Shen,J.; Kehtarnavaz,N.;Shah,M.;, "Deep Learning-Based Human Pose Estimation: A Survey," p. 18, 2 January 2021.

[21] Neurohive, "Inferring a 3D Human Pose out of a 2D Image with FBI," 16 July 2018. [Online]. Available: https://neurohive.io/en/state-of-the-art/inferring-a-3d-human-pose-out-of-a-2d-image-with-fbi/.

[22] Bazarevsky,V.; Grishchenko,I.; Raveendran,K.;Zhu,T.; Zhang,F.; Grundmann,M.;, "Towards Multi-Person Pose Tracking : Bottom-up and Top-down Methods," p. 1, 2017.

[23] Bazarevsky,V.;Grishchenko,I.;Raveendran,K.;Zhu,T.;Zhang,F.;Grundmann,M.;, "BlazePose: On-device Real-time Body Pose tracking," p. 1, 17 June 2020.

[24] MediaPipe, "MediaPipe Pose," [Online]. Available: https://google.github.io/mediapipe/solutions/pose.html.

[25] Wikipedia, "Raspberry Pi," 14 October 2021. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi.

[26] R. Pi. [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.html#gpio-and-the-40-pin-header.

[27] smartbitbn, "Raspberry Pi 4 Model B – 2GB," 18 October 2021. [Online]. Available: https://smartbitbn.com/product/raspberry-pi-4-model-b-2gb/.

[28] Aquarium Article Digest, 11 April 2019. [Online]. Available: https://aquarium-digest.com/2010/04/.

[29] Babul Singh, 31 July 2020. [Online]. Available: https://primefeed.in/news/3661440/optical-sensing-market-research-with-covid-19-rohm-semiconductor-abb-hamamatsu-photonics/.

[30] Ai-Thinker, October 18 2021. [Online]. Available: https://www.linkedin.com/company/ai-thinker.

[31] Ai-Thinker, "Pudding series development board-A9G development board information," 18 October 2021. [Online]. Available: https://docs.ai-thinker.com/en/gprs/a9g/boards.

[32] DFROBOT, "SKU:TEL0134," 18 October 2021. [Online]. Available: https://wiki.dfrobot.com/A9G_Module_SKU_TEL0134.

[33] L. Hattersley, "The MagPi," 2019. [Online]. Available: https://magpi.raspberrypi.org/articles/set-up-raspberry-pi-4. [Accessed 12 September 2021].

[34] Wikipedia, "Raspberry Pi OS," 6 September 2021. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi_OS. [Accessed 25 September 2021].

[36] Wikipedia, "OpenCV," 7 August 2021. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV. [Accessed 22 September 2021].

[37] A. Rosebrock, 2 February 2015. [Online]. Available: https://www.pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions/. [Accessed 22 September 2021].

[38] Google, "MediaPipe," [Online]. Available: https://opensource.google/projects/mediapipe. [Accessed 5 October 2021].

[39] Wikipedia, "NumPy," 23 August 2021. [Online]. Available: https://en.wikipedia.org/wiki/NumPy. [Accessed 22 September 2021].

[40] OpenCV, 5 July 2021. [Online]. Available: https://docs.opencv.org/3.4.15/d6/d0f/group__dnn.html#ga29d0ea5e52b1d1a6c2681e3f7d68473a.

[41] OpenCV, 5 July 2021. [Online]. Available: https://docs.opencv.org/3.4.15/d6/d0f/group__dnn.html#ga29f34df9376379a603acd8df581ac8d7.

[42] G. Zhong, 19 December 2019. [Online]. Available: https://towardsdatascience.com/drowsiness-detection-with-machine-learning-765a16ca208a.