ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και υλοποίηση ενός ενσωματωμένου συστήματος με δυνατότητες τεχνητής νοημοσύνης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Πασχάλη Μαργαρίτη

Επιβλέπων: Δρ. Δασυγένης Μηνάς

Αναπληρωτής Καθηγητής

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

ΚΟΖΑΝΗ/ΙΟΥΛΙΟΣ/2025

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL &
COMPUTER ENGINEERING

Design and implementation of an embedded system with artificial intelligence capabilities

DIPLOMA THESIS

Pashalis Margaritis

SUPERVISOR: Dr. Minas Dasygenis

Associate Professor

Robotics, Embedded and Integrated Systems Laboratory

KOZANI/JULY/2025

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο " Σχεδιασμός και υλοποίηση ενός ενσωματωμένου συστήματος με δυνατότητες τεχνητής νοημοσύνης " καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Δασυγένη Μηνά, Αναπληρωτή Καθηγητή αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: 

Περίληψη

Η ένταξη της Τεχνητής Νοημοσύνης (ΤΝ) σε ενσωματωμένα συστήματα αποτελεί ένα σημαντικό βήμα προς τη ανάπτυξη έξυπνων, αυτόνομων και ενεργειακά αποδοτικών λύσεων, με εφαρμογές σε ένα ευρύ φάσμα τομέων, όπως η βιομηχανία, η υγεία, οι έξυπνες πόλεις και οι καταναλωτικές συσκευές. Η παρούσα εργασία παρουσιάζει τον σχεδιασμό και την υλοποίηση ενός πρωτοτύπου ενσωματωμένου συστήματος με δυνατότητες Τεχνητής Νοημοσύνης, το οποίο συνδυάζει πολλαπλές λειτουργίες, όπως Διαδικτυακό Ραδιόφωνο (Internet Radio), Φωτογραφική Μηχανή (Camera) και Υπολογιστική Όραση (Computer Vision). Η συσκευή βασίζεται στην πλατφόρμα ESP32, η οποία ενσωματώνει έναν χαμηλής κατανάλωσης μικροελεγκτή, προσφέροντας έτσι μια ιδανική ισορροπία μεταξύ ενέργειας και υψηλής απόδοσης υπολογιστικής ισχύος. Για τη λήψη φωτογραφιών χρησιμοποιείται η κάμερα, ενώ οι δυνατότητες Τεχνητής Νοημοσύνης αξιοποιούνται μέσω υπηρεσιών cloud του Microsoft Azure, και πιο συγκεκριμένα του Computer Vision API (Application Programming Interface), που επιτρέπει την αναγνώριση και επεξεργασία εικόνας σε πραγματικό χρόνο. Ο προγραμματισμός της πλατφόρμας έγινε με ένα μοντέρνο πρόγραμμα επεξεργασίας κώδικα το Arduino IDE 2.3.5 (Integrated Development Environment), χρησιμοποιώντας την γλώσσα C. Το πρωτότυπο αποτελεί απόδειξη ότι η ΤΝ μπορεί ενσωματωθεί σε συστήματα χαμηλού κόστους, χωρίς να θυσιάζεται η αξιοπιστία. Παράλληλα, η επιτυχής υλοποίηση της εφαρμογή υπογραμμίζει τη σημασία της σύγχρονης τεχνολογίας (όπως το ESP32 και οι υπηρεσίες cloud) στη δημιουργία έξυπνων συσκευών, προσφέροντας ένα πλαίσιο για μελλοντικές επεκτάσεις, όπως η βελτίωση των αλγορίθμων AI ή η εισαγωγή νέων λειτουργιών (π.χ., φωνητικής διεπαφής).

Λέξεις Κλειδιά: Μικροελεγκτής, Ενσωματωμένα Συστήματα, Τεχνητή Νοημοσύνη, Υπολογιστικό νέφος, Διαδικτυακό Ραδιόφωνο, υπολογιστική όραση, Φωτογραφική Μηχανή, ενεργειακή απόδοση, πλατφόρμα ESP32, Arduino IDE

Abstract

The integration of Artificial Intelligence (AI) into embedded systems represents a significant step toward the development of smart, autonomous, and energy-efficient solutions, with applications across a wide range of fields such as industry, healthcare, smart cities, and consumer devices. This work presents the design and implementation of a prototype embedded system with AI capabilities, combining multiple functionalities including Internet Radio, a Camera, and Computer Vision. The device is based on the ESP32 platform, which incorporates a low-power microcontroller, thus offering an ideal balance between energy efficiency and high computational performance. For image capture, a camera module is used, while AI capabilities are enabled through Microsoft Azure cloud services, specifically the Computer Vision API, which allows real-time image recognition and processing. The ESP32 platform was programmed using a modern code editor, Arduino IDE 2.3.5, and the C programming language. The prototype serves as proof that AI can be integrated into low-cost systems without compromising reliability. At the same time, the successful implementation highlights the importance of modern technologies—such as ESP32 and cloud services—in the development of smart devices, providing a framework for future expansions, such as improving AI algorithms or introducing new features (e.g., voice interface).

Keywords: Microcontroller, Embedded Systems, Artificial Intelligence, Cloud, Internet Radio, Computer Vision, Camera, Energy Efficiency, ESP32 Platform, Arduino IDE

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του προπτυχιακού προγράμματος σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Δυτικής Μακεδονίας. Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη σε όλους όσοι συνέβαλαν ουσιαστικά στην ολοκλήρωσή της. Ιδιαίτερα, εκτιμώ τη γνώση και τις δεξιότητες που αποκόμισα από τους διδάσκοντες του τμήματος, οι οποίοι διαδραμάτισαν καθοριστικό ρόλο στην ακαδημαϊκή και επαγγελματική μου εξέλιξη στον τομέα της Μηχανικής Υπολογιστών και των Ηλεκτρονικών Συστημάτων.

Θα ήθελα να εκφράσω τις ιδιαίτερες ευχαριστίες μου στον επιβλέποντα της παρούσας εργασίας, Δρ. Μηνά Δασυγένη, για την εμπιστοσύνη που έδειξε αναθέτοντάς μου το συγκεκριμένο θέμα, αλλά και για την αδιάκοπη υποστήριξη και την πολύτιμη καθοδήγησή του καθ' όλη τη διάρκεια της ερευνητικής διαδικασίας. Η συμβολή του υπήρξε καθοριστική για την επίτευξη των στόχων της διπλωματικής εργασίας.

Παράλληλα, θα ήθελα να εκφράσω την ειλικρινή μου εκτίμηση προς τα μέλη του Εργαστηρίου Ρομποτικής και Ενσωματωμένων Συστημάτων, και ιδιαίτερα προς τους διδακτορικούς φοιτητές και τους συμφοιτητές μου, για την τεχνική υποστήριξη, τις δημιουργικές συζητήσεις και την πολύτιμη πρόσβαση στις πειραματικές υποδομές, στοιχεία που ήταν απαραίτητα για την υλοποίηση του πρακτικού μέρους της εργασίας.

Τέλος, νιώθω την ανάγκη να ευχαριστήσω θερμά την οικογένειά μου και τους φίλους μου, τόσο από τον ακαδημαϊκό χώρο όσο και εκτός αυτού, για την αμέριστη αγάπη, την κατανόηση και τη συνεχή ηθική στήριξη που μου προσέφεραν. Η συμβολή τους υπήρξε καθοριστική, τόσο σε συναισθηματικό όσο και σε πρακτικό επίπεδο, καθιστώντας δυνατή την ολοκλήρωση αυτής της εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Πρόλογος

Στις επόμενες σελίδες παρουσιάζεται αναλυτικά η έρευνα και η ανάπτυξη ενός φορητού ενσωματωμένου συστήματος βασισμένου σε ESP32, το οποίο ενσωματώνει υπηρεσίες Τεχνητής Νοημοσύνης μέσω Cloud. Το προγραμματιστικό μέρος της παρούσας διπλωματικής εργασίας υλοποιήθηκε με τη χρήση του λογισμικού Arduino IDE, το οποίο διατίθεται δωρεάν από την εταιρεία Arduino. Η έρευνα πραγματοποιήθηκε στο Εργαστήριο Ρομποτικής και Ενσωματωμένων Συστημάτων του Πανεπιστημίου Δυτικής Μακεδονίας (Π.Δ.Μ.), το οποίο εδρεύει στην πόλη της Κοζάνης. Η επιτυχής ολοκλήρωση της συγκεκριμένης μελέτης, καθώς και η διεξαγωγή των πειραμάτων της, δεν θα μπορούσε να πραγματοποιηθεί χωρίς την καθοδήγηση, επίβλεψη και στήριξη από τα μέλη του εργαστηριακού προσωπικού, αλλά και τον επιβλέποντα καθηγητή μου. Η συνεισφορά τους υπήρξε καθοριστική για την ανάπτυξη και την υλοποίηση της εργασίας.

Κεφάλαιο 1:Εισαγωγή

Για να εξασφαλιστεί η πληρότητα του κεφαλαίου, περιλαμβάνεται μια σύντομη περιγραφή των ενσωματωμένων συστημάτων και της χρήσης τους. Τέλος, γίνεται μια σύγκριση με προ υπάρχουσες υλοποιήσεις του εμπορίου και διατυπώνεται ο βασικός στόχος της διπλωματικής εργασίας,. Η δομή αυτή αποσκοπεί στη δημιουργία ενός συνεκτικού πλαισίου αναφοράς για τη συνέχεια της έρευνας.

1.1 Αντικείμενο της διπλωματικής

Αντικείμενο της παρούσας διπλωματικής εργασίας αποτελεί η μελέτη, ο σχεδιασμός και η υλοποίηση ενός ενσωματωμένου (Embedded) συστήματος με δυνατότητες τεχνητής νοημοσύνης (Artificial intelligence) μέσω Cloud, με έμφαση στην υπολογιστική όραση (Computer vision), αξιοποιώντας υπηρεσίες και εργαλεία της πλατφόρμας Microsoft Azure. Η εργασία περιλαμβάνει την ανάπτυξη μιας ολοκληρωμένης λύσης, βασισμένης σε κατάλληλο υλικό (ESP32 και ESP32-CAM), το οποίο ενσωματώνει αισθητήρες εικόνας (π.χ. κάμερα) και μπορεί να επεξεργάζεται δεδομένα εικόνας μέσω του Cloud.

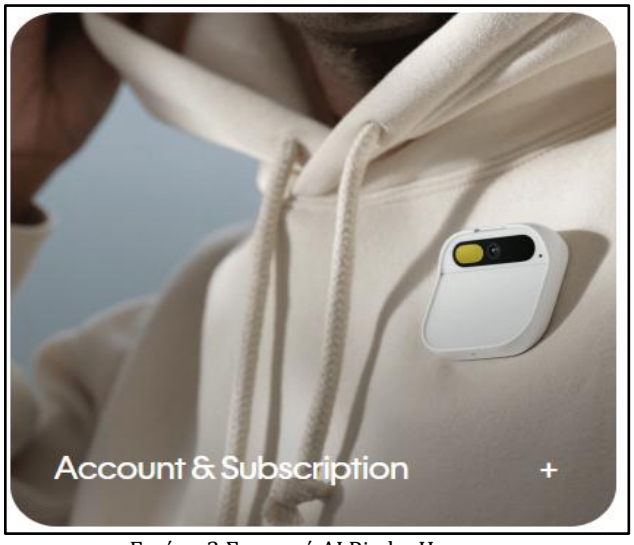
Η τεχνητή νοημοσύνη αξιοποιείται για την αναγνώριση κυρίως αντικειμένων, αλλά και προσώπων σε πραγματικό χρόνο, ενώ το Azure χρησιμοποιείται τόσο για την ανάπτυξη και εκπαίδευση των μοντέλων (μέσω υπηρεσιών όπως το Azure Vision Studio), όσο και για την αποθήκευση και επεξεργασία των δεδομένων και φωτογραφιών. Επιπλέον, υλοποιείται η διασύνδεση του ενσωματωμένου συστήματος με cloud-based dashboards (Azure Storage Account και containers).

1.2 Παρόμοια έρευνα στο αντικείμενο της διπλωματικής

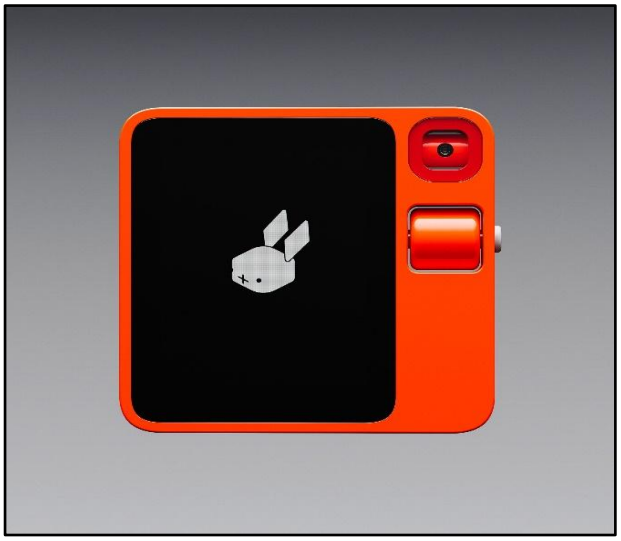
Η συσκευή που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας συγκρίνεται με τέσσερα πρόσφατα και καινοτόμα προϊόντα της αγοράς: το Ray-Ban Meta [1], Limitless AI Pendant [2], το Rabbit R1 (Rabbit Pocket Companion) [3] και το AI Pin by Humane [4]. Παρότι τα εμπορικά αυτά προϊόντα προορίζονται κυρίως για καθημερινή χρήση από καταναλωτές και διατίθενται στην αγορά με έμφαση στην εργονομία, τη φορητότητα και την ενσωμάτωση εξελιγμένων τεχνολογιών, παρουσιάζουν κοινά σημεία με τη συσκευή της διπλωματικής, τόσο ως προς τη φιλοσοφία όσο και τις τεχνολογικές δυνατότητες, κάποιες από αυτές είναι η αξιοποίηση τεχνητής νοημοσύνης, η ενσωμάτωση κάμερας και η παροχή έξυπνων υπηρεσιών (βλπ. Εικόνες 1,2,3 και 4).

Εικόνα 1:Meta Ray-Ban Smart Glasses
Πηγή: https://www.meta.com/ai-glasses/shop-all/

Εικόνα 2:Limitless AI Pendant
Πηγή:https://www.limitless.ai/



Εικόνα 3:Συσκευή AI Pin by Human
Πηγή: https://humane.com/Humane Ai Pin



Εικόνα 4:Rabbit R1 by Rabbit Pocket Companion
Πηγή: https://www.rabbit.tech/rabbit-r1

Παρακάτω παρατίθεται ένας συγκριτικός πίνακας που παρουσιάζει τις τεχνικές προδιαγραφές και τα χαρακτηριστικά της συσκευής που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, σε σύγκριση με τα εμπορικά προϊόντα Rabbit R1 [4], Humane AI Pin [3], Ray-Ban Meta [1] και Limitless AI Pendant [2] (βλπ. Πίνακας 1).

Πίνακας 1:Συγκριτικός πίνακας τεχνικών προδιαγραφών

Χαρακτηριστικά	Συσκευή διπλωματικής	Rabbit R1	Humane AI Pin	Meta Ray-Ban	Limitless Pendant
Επεξεργαστής (CPU)	ESP32 (Xtensa LX6 dual-core)	MediaTek Helio P35 (MT6765) έως 2.3GHz	Qualcomm Snapdragon 8-core, 2.1GHz	ARM Cortex-M4	Qualcomm Snapdragon Wear
Μνήμη (RAM)	520 KB SRAM (+ PSRAM για ESP32 CAM)	4GB	4GB	256MB	1GB
Αποθηκευτικός Χώρος	Δεν καθορίζεται	128GB	32GB	256MB	1GB
Οθόνη	3.2" TFT οθόνη αφής	2.88" TFT οθόνη αφής	Laser Ink Display (χωρίς οθόνη)	Χωρίς οθόνη	Χωρίς οθόνη (ακουστικά)
Κάμερα	OV2640 (2MP)	8MP (νυχτερινή όραση)	12MP, 1080p@30FPS	Χωρίς κάμερα	12MP (βίντεο 1080p)
Συνδεσιμότητα	Wi-Fi	Wi-Fi, Bluetooth 5.0, 4G LTE	Wi-Fi, LTE (T-Mobile), Bluetooth	Wi-Fi, Bluetooth	Bluetooth, Wi-Fi (μέσω app)
Αισθητήρες	Δεν καθορίζεται	Μαγνητόμετρο,GPS, Επιταχυνσιόμετρο, Γυροσκό	Κάμερα, Επιταχυνσιόμετρο, Γυροσκόπιο, Βάθος/Κίνηση	Μικρόφωνο, επιταχυνσιόμετρο	επιταχυνσιόμετρο, γυροσκόπιο, μικρόφωνο
Ήχος	Μονοφωνικό ηχείο	Διπλό μικρόφωνο + ηχείο	Personic speaker + Bluetooth ακουστικά	Μικρόφωνο, ηχείο	Ηχείο, μικρόφωνο
Διαστάσεις (MxΠxΥ)	115mm x 67mm x 57mm	78mm x 78mm x 13mm	44.5mm x 47.5mm x 14.98mm	40x40x10 mm	150x50x15 mm
Βάρος	225g	115g	54.7g (συσκευή + μπαταρία)	20g	50g
Μπαταρία	2200mAh	1000mAh	140mAh + πρόσθετη μπαταρία	150mAh (standby 100+ ώρες)	500mAh (6 ώρες)
Λειτουρ. Σύστημα	Δεν καθορίζεται	Rabbit OS (Android-based)	CosmOS	Custom RTOS	Custom OS (Android-based)
Διασύνδ. Χρήστη	Οθόνη αφής	Φωνή, Οθόνη, Τροχός, Κουμπί	Φωνή, Χειρονομίες, Laser προβολή	Φωνή	Φωνή, αφή πλαισίου
Τιμή	$45 (5,000 δωρεάν συναλλαγές/ μήνα)	$199	$199	$699+$24/μήνα	$299+

Αν συγκρίνουμε τις πέντε συσκευές, δηλαδή τη συσκευή της διπλωματικής, το Rabbit R1 [4], το Humane AI Pin [3], το Ray-Ban Meta Smart Glasses [1] και το Limitless AI Pendant [2], διαπιστώνουμε πως καθεμία προσεγγίζει με διαφορετικό τρόπο τον σχεδιασμό έξυπνων και φορητών τεχνολογιών. Η συσκευή της διπλωματικής ξεχωρίζει για τη χαμηλή κατανάλωση και το προσιτό κόστος, καθώς χρησιμοποιεί δυο μικροελεγκτές ESP32 και ESP32 CAM και συνδέεται με τις υπηρεσίες cloud της Microsoft Azure για τεχνητή νοημοσύνη. Υποστηρίζει επίσης λειτουργίες όπως διαδικτυακό ραδιόφωνο και αναγνώριση εικόνας. Από την άλλη, το Rabbit R1 προσφέρει μεγαλύτερη υπολογιστική ικανότητα και μνήμη, διαθέτει ενσωματωμένες κάμερες υψηλής ανάλυσης και GPS. Το Humane AI Pin απευθύνεται σε πιο premium χρήστες, με προηγμένους αισθητήρες, φωνητικές εντολές, laser projector και με συνδρομή. Το Ray-Ban Meta ενσωματώνει έξυπνες λειτουργίες σε μια καθημερινή και λειτουργική μορφή, όπως είναι τα γυαλιά. Με ενσωματωμένη κάμερα, ηχεία, υποστήριξη hands-free λειτουργιών, και δυνατότητα λήψης και αποστολής περιεχομένου, στοχεύει να κάνει την καθημερινότητα πιο εύκολη. Και τέλος, το Limitless AI Pendant εστιάζει κυρίως στην καταγραφή συνομιλιών και στην προσωποποιημένη υποστήριξη AI. Ενώ οι εμπορικές λύσεις προσφέρουν εξελιγμένο hardware και περισσότερη φορητότητα, η συσκευή της διπλωματικής δίνει έμφαση στη απλότητα λειτουργίας, την εκπαιδευτική αξία και την ευελιξία, αποτελώντας μια άξια και οικονομική επιλογή για την ενσωμάτωση ΤΝ σε ενσωματωμένα συστήματα.

1.3 Στόχος της διπλωματικής

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο τη δημιουργία ενός φορητού και αποδοτικού ενσωματωμένου συστήματος, του οποίου η βασική λειτουργία επικεντρώνεται στην υπολογιστική όραση. Παράλληλα, υποστηρίζονται επιμέρους δυνατότητες όπως η αναπαραγωγή Internet Radio και η χρήση κάμερας για λήψη και αποθήκευση φωτογραφιών. Επίσης, έμφαση δίνεται και στον αποτελεσματικό σχεδιασμό του λογισμικού, το οποίο είναι απαλλαγμένο από περιττές ή ανεπιθύμητες εφαρμογές (bloatware), διασφαλίζοντας έτσι ταχύτητα, αξιοπιστία και πλήρη αξιοποίηση των διαθέσιμων πόρων του συστήματος. Για την επίτευξη των προαναφερόμενων δυνατοτήτων, ως προγραμματιζόμενη πλακέτα επιλέχτηκε η πλακέτα ESP32 [5] και ESP32 CAM [6][7], της εταιρίας Espressif Systems, η οποία διαθέτει τις κατάλληλες δυνατότητες προγραμματισμού για τον επεξεργαστή της και υποστηρίζεται ευρέως τόσο από προγράμματα και βιβλιοθήκες. Το SOC (System-on-Chip) που χρησιμοποιείται είναι σχεδιασμένο για απλότητα λόγω της ολοκληρωμένης του δομής, περιλαμβάνοντας όλα τα βασικά στοιχεία όπως επεξεργαστή Xtensa LX6, μνήμη SRAM, Rom, κάμερα OV2640 2MP [8], WiFi(Wireless Fidelity) 802.11 b/g/n, Bluetooth/BLE και Bootloader. Ο επεξεργαστής της πλακέτας είναι ένας 32-bit dual-core RISC(Reduced Instruction Set Computer) επεξεργαστής υψηλής απόδοσης και χαμηλής κατανάλωσης. κατασκευασμένος με τεχνολογία 40nm. Λεπτομερείς πληροφορίες σχετικά με την πλακέτα θα παρουσιαστούν σε επόμενο κεφάλαιο. Για τον προγραμματισμό και για τις δοκιμές χρησιμοποιήθηκε το εξειδικευμένο περιβάλλον του Arduino IDE, ενώ για την κατασκευή του ενσωματωμένου συστήματος, το Autodesk Fusion 360, EasyEDA και Creality Print.

1.4 Οργάνωση της διπλωματικής

Προκειμένου να διασαφηνιστεί το θέμα της παρούσας διπλωματικής εργασίας, είναι απαραίτητο να καθορίσουμε το δομικό πλαίσιο της μελέτης. Η παρούσα μελέτη οργανώθηκε σε πέντε κεφάλαια, που ακολουθούν μια ιεραρχική προσέγγιση από τη θεωρία προς την πράξη, με στόχο τη συστηματική ανάλυση και υλοποίηση ενός φορητού ενσωματωμένου συστήματος με δυνατότητες τεχνητής νοημοσύνης.

Στο Κεφάλαιο 2, με τίτλο «Θεωρητικό Υπόβαθρο», παρουσιάζονται ενδελεχώς τα ενσωματωμένα συστήματα και τα απαραίτητα πρωτόκολλα επικοινωνίας, αναλύοντας τις δυνατότητες και τους περιορισμούς που αντιμετωπίστηκαν. Επιπλέον, γίνεται αναφορά στα λογισμικά που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας.

Στη συνέχεια, στο Κεφάλαιο 3 περιγράφεται η διαδικασία σχεδίασης και κατασκευής της ενσωματωμένης συσκευής που αναπτύχθηκε στο πλαίσιο της διπλωματικής. Ιδιαίτερη έμφαση δίνεται στην επιλογή και ενσωμάτωση των ηλεκτρονικών εξαρτημάτων, όπως ο μικροελεγκτής ESP32, ο ψηφιακός ενισχυτής ήχου, η έγχρωμη οθόνη αφής και άλλα περιφερειακά. Παρουσιάζονται τα τεχνικά χαρακτηριστικά κάθε εξαρτήματος, η μεθοδολογία σύνδεσης και ολοκλήρωσης του κυκλώματος, καθώς και η χρήση εργαλείων CAD και 3D εκτύπωσης για την κατασκευή του περιβλήματος. Ο στόχος είναι η πλήρης τεκμηρίωση της μετάβασης από τη θεωρητική σχεδίαση στο λειτουργικό πρωτότυπο.

Στο Κεφάλαιο 4 γίνεται αναλυτική παρουσίαση των λογισμικών εργαλείων που αξιοποιήθηκαν για τον προγραμματισμό και την υλοποίηση εφαρμογών με τους μικροελεγκτές ESP32 και ESP32-CAM. Παρουσιάζεται η αρχιτεκτονική τους, καθώς και η διάρθρωση των βασικών συναρτήσεων που σχετίζονται με τη λειτουργικότητα της συσκευής.

Στο Κεφάλαιο 5 πραγματοποιείται αναλυτική παρουσίαση και αξιολόγηση των αποτελεσμάτων που προέκυψαν από την υλοποίηση του φορητού ενσωματωμένου συστήματος. Εξετάζονται διεξοδικά τόσο οι επιδόσεις όσο και οι περιορισμοί της συσκευής, με στόχο την ανάδειξη πιθανών τομέων βελτιστοποίησης. Τα βασικά συμπεράσματα που προέκυψαν επιβεβαιώνουν την επιτυχία της υλοποίησης, ενώ ταυτόχρονα προτείνονται σαφείς κατευθύνσεις για μελλοντική εξέλιξη του συστήματος, με έμφαση στη βελτίωση της αποδοτικότητας και της επεκτασιμότητάς του.

Κεφάλαιο 2:Θεωρητικό Υπόβαθρο

2.1 Ενσωματωμένα συστήματα

Τα ενσωματωμένα συστήματα είναι εξειδικευμένοι υπολογιστές οι οποίοι συνδυάζουν έναν επεξεργαστή, μνήμη και περιφερειακές μονάδες για να εκτελούν συγκεκριμένες λειτουργίες μέσα σε ένα μεγαλύτερο ηλεκτρονικό ή μηχανικό σύστημα. Ενσωματώνονται πλήρως στο τελικό προϊόν, το οποίο συνήθως αποτελείται από ηλεκτρονικά εξαρτήματα και μηχανικά μέρη. Η φύση αυτών των συστημάτων σημαίνει ότι, σε πολλές περιπτώσεις, πρέπει να υποστηρίζουν λειτουργίες σε πραγματικό χρόνο, καθώς ελέγχουν άμεσα τη φυσική συμπεριφορά του συστήματος στο οποίο ανήκουν. Η σημασία της ένταξης ενσωματωμένων συστημάτων αυξάνεται συνεχώς καθώς βρίσκονται σε πληθώρα συσκευών που χρησιμοποιούμε καθημερινά από οικιακές συσκευές και φορητές συσκευές, έως βιομηχανικές, ρομποτικές, μέσα μεταφοράς, ακόμη και ιατρικές συσκευές. Σύμφωνα με έρευνα [9], η αγορά αναμένεται να αυξηθεί από 101,44 δισεκατομμύρια δολάρια το 2023 σε 110,89 δισεκατομμύρια δολάρια το 2024, με σύνθετο ετήσιο ρυθμό ανάπτυξης (CAGR) 9,3% και προβλέπεται ότι η αγορά θα συνεχίσει να αναπτύσσεται, φτάνοντας τα 159,1 δισεκατομμύρια δολάρια έως το 2028, με CAGR 9,4%, πράγμα που δείχνει πόσο σημαντικό ρόλο παίζουν στην πρόοδο της τεχνολογίας. Η βάση των σύγχρονων ενσωματωμένων συστημάτων είναι οι μικροελεγκτές, οι οποίοι αξιοποιούνται για απλές εργασίες, σε πιο περίπλοκες εφαρμογές, χρησιμοποιούνται υπολογιστές υλοποιημένοι σε μια πλακέτα (SBC - Single-Board Computer), ενώ σε εξαιρετικά εξειδικευμένες λειτουργείες όπως κρυπτογράφηση, χρησιμοποιείται η διάταξη προγραμματιζόμενων πυλών στο πεδίο, γνωστή ως FPGA (Field-Programmable Gate Array).

2.1.1 Μικροελεγκτές:Ορισμός και λειτουργία

Ο μικροελεγκτής (microcontroller) είναι ένα ολοκληρωμένο κύκλωμα σχεδιασμένο να ελέγχει συγκεκριμένες λειτουργίες σε ενσωματωμένα συστήματα. Είναι εξειδικευμένοι μικρο-υπολογιστές χωρίς πολύπλοκα λειτουργικά συστήματα, σχεδιασμένοι να διευθύνουν συγκεκριμένες διεργασίες. Σε ένα ενιαίο τσιπ περιλαμβάνονται:

- **Κεντρικό Επεξεργαστή (CPU)**. Η CPU (Central Processing Unit), γνωστή και ως επεξεργαστής, αποτελεί τον "εγκέφαλο" του μικροελεγκτή. Είναι υπεύθυνη για την επεξεργασία και εκτέλεση των εντολών που καθορίζουν τη λειτουργία του συστήματος. Συγκεκριμένα, εκτελεί αριθμητικές και λογικές πράξεις (π.χ. πρόσθεση, σύγκριση δεδομένων), διαχειρίζεται λειτουργίες εισόδου/εξόδου (I/O) για αλληλεπίδραση με το περιβάλλον και τέλος πραγματοποιεί λειτουργίες μεταφοράς δεδομένων, μεταφέροντας οδηγίες σε άλλα στοιχεία του ενσωματωμένου συστήματος.

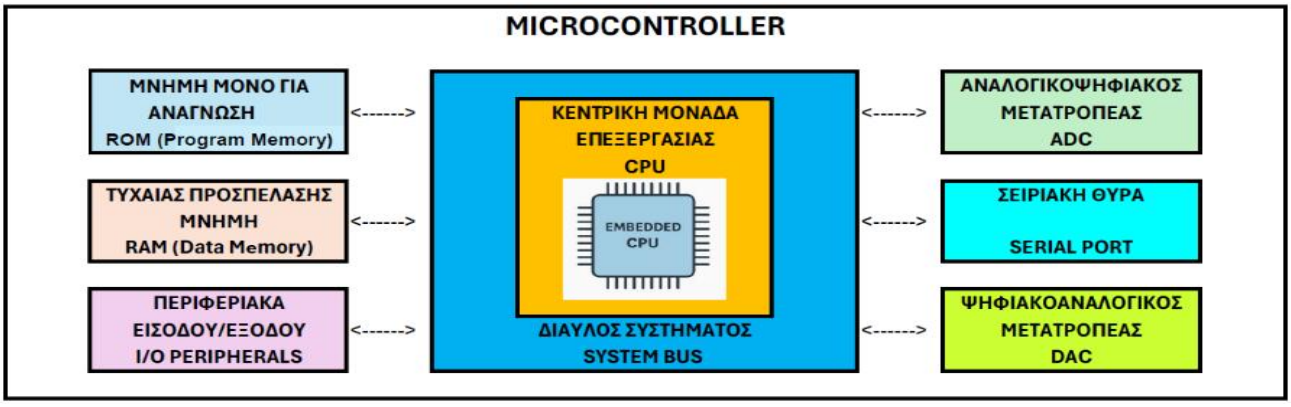
- **Μνήμη (προγράμματος και δεδομένων)**. Η μνήμη του αποθηκεύει δεδομένα και εντολές που χρησιμοποιούνται από την CPU. Διακρίνεται σε δύο τύπους, στη μνήμη προγράμματος (Program Memory), που αποθηκεύει μακροπρόθεσμες εντολές (κώδικα) που εκτελεί η CPU και είναι μη πτητική (non-volatile), δηλαδή διατηρεί τα δεδομένα ακόμα και χωρίς τροφοδοσία (π.χ., Flash μνήμη). Και στη μνήμη δεδομένων (Data Memory), η οποία χρησιμοποιείται για προσωρινή αποθήκευση δεδομένων κατά την εκτέλεση εντολών, είναι πτητική (volatile), δηλαδή χάνει τα δεδομένα όταν απενεργοποιηθεί η τροφοδοσία (π.χ. RAM).

- **Περιφερειακά εισόδου/εξόδου (I/O) (π.χ., GPIO, ADC, UART)**. Τα περιφερειακά I/O λειτουργούν ως διεπαφή μεταξύ της CPU και του εξωτερικού κόσμου, χωρίζονται σε δυο κατηγορίες, στις θύρες εισόδου (Input Ports), οποίες λαμβάνουν πληροφορίες από εξωτερικές συσκευές (π.χ., αισθητήρες) και τις μεταφέρουν στην CPU ως δυαδικά δεδομένα (0s και 1s) και στις θύρες εξόδου (Output Ports), που εκτελούν ενέργειες με βάση τις οδηγίες της CPU, ελέγχοντας εξωτερικές συσκευές (π.χ., κινητήρες, οθόνες).

Εκτός από τα βασικά δομικά στοιχεία ενός μικροελεγκτή, υπάρχουν και επιπλέον στοιχεία που υποστηρίζουν τις λειτουργείες του, τέτοια δομικά στοιχεία είναι:

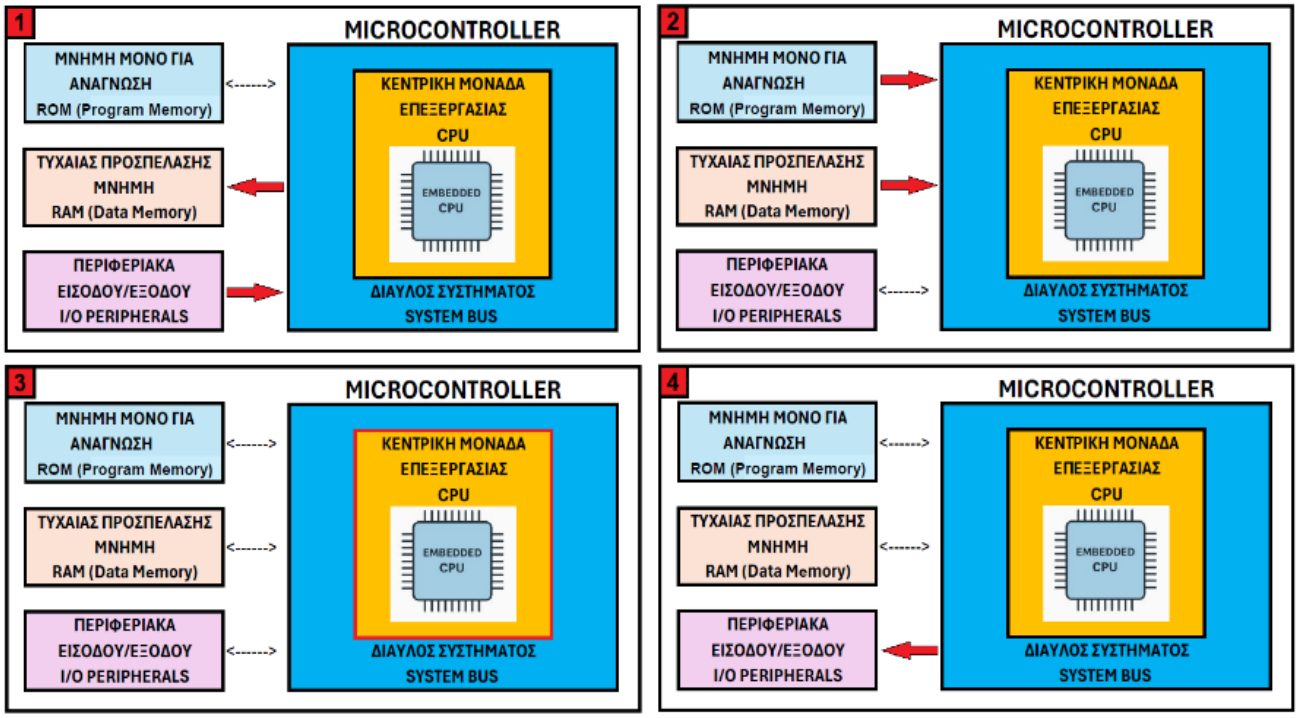
- **Αναλογο-ψηφιακός μετατροπέας (ADC):** Ο ADC (Analog-to-Digital Converter) είναι ένα κύκλωμα που μετατρέπει αναλογικά σήματα σε ψηφιακά, επιτρέποντας στον επεξεργαστή του μικροελεγκτή να αλληλεπιδρά με εξωτερικές αναλογικές συσκευές, όπως αισθητήρες (π.χ. θερμοκρασίας ή φωτός).
- **Ψηφιακο-αναλογικός μετατροπέας (DAC):** Ο DAC (Digital-to-Analog Converter) εκτελεί την αντίστροφη λειτουργία του ADC, μετατρέποντας ψηφιακά σήματα σε αναλογικά. Με τον τρόπο αυτό, ο μικροελεγκτής μπορεί να ελέγχει εξωτερικές αναλογικές συσκευές, όπως κινητήρες ή οθόνες.
- **Δίαυλος συστήματος (System bus)**. Ο δίαυλος συστήματος είναι ο κεντρικός αγωγός επικοινωνίας που συνδέει όλα τα εσωτερικά στοιχεία του μικροελεγκτή (CPU, μνήμη, I/O), διασφαλίζοντας τη ροή δεδομένων και εντολών.
- **Σειριακή θύρα (Serial port)**. Η σειριακή θύρα είναι ένας τύπος περιφερειακού I/O που διευκολύνει τη σύνδεση του μικροελεγκτή με εξωτερικές συσκευές. Λειτουργεί με διαδοχική (σειριακή) ανταλλαγή bits, σε αντίθεση με παράλληλες θύρες ή USB(Universal Serial Bus), οι οποίες μεταφέρουν πολλά bits ταυτόχρονα.

Στο παρακάτω σχήμα (βλπ Σχήμα 1) παρουσιάζεται η αρχιτεκτονική, μαζί με τα βασικά στοιχεία και τις διασυνδέσεις, από τα οποία αποτελείται ένας μικροελεγκτής.



Σχήμα 1:Παράδειγμα αρχιτεκτονικής ενός μικροελεγκτή:Βασικά στοιχεία και διασυνδέσεις

Αφού παρουσιάστηκε η αρχιτεκτονική, ακολουθεί η ανάλυση και περιγραφή της αρχής λειτουργίας ενός μικροελεγκτή. Σε πρώτο στάδιο πραγματοποιείται η λήψη δεδομένων, όπου οι πληροφορίες συλλέγονται μέσω των περιφερειακών I/O (π.χ. αισθητήρες, κουμπιά). Στη συνέχεια, τα δεδομένα αποθηκεύονται προσωρινά στη μνήμη δεδομένων (RAM). Σε δεύτερο στάδιο, ο επεξεργαστής λαμβάνει τα δεδομένα από τη RAM και προχωρά στην επεξεργασία και ερμηνεία τους, σύμφωνα με τις εντολές που είναι αποθηκευμένες στη μνήμη προγράμματος (ROM/Flash). Τέλος, μέσω των διεπαφών εισόδου/εξόδου (I/O), ο μικροελεγκτής ενεργοποιεί τις εξωτερικές συσκευές (π.χ. κινητήρες,οθόνες). Στο παρακάτω σχήμα αποτυπώνεται αναλυτικά η αρχή λειτουργίας ενός μικροελεγκτή (βλ. Σχήμα 2).



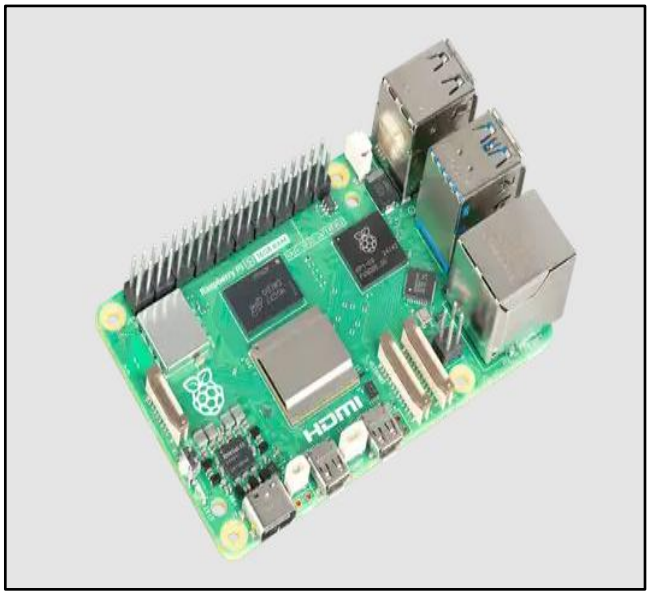
Σχήμα 2:Αρχή λειτουργίας ενός μικροελεγκτή – Βήματα 1,2,3,4

Τα κύρια χαρακτηριστικά περιγράφονται παρακάτω και είναι, η χαμηλή κατανάλωση ιδανικό για συσκευές με μπαταρίες, ο πραγματικός Χρόνος (Real-Time), άμεση απόκριση σε εξωτερικά γεγονότα και τέλος η οικονομία, που επιτρέπει χαμηλό κόστος μαζικής παραγωγής. Ένα παράδειγμα που γίνεται χρήση μικροελεγκτών, οι οποίοι συνεργάζονται για τον έλεγχο διαφορετικών υποσυστημάτων σε πιο πολύπλοκα συστήματα, είναι σε ένα αυτοκίνητο, οπού το σύστημα αντιμπλοκαρίσματος φρένων (ABS - Antiblock Breaking System), το σύστημα ελέγχου πρόσφυσης (traction control) και του ψεκασμού καυσίμου, επικοινωνούν μεταξύ τους ή με ένα κεντρικό υπολογιστή, ανταλλάσσοντας δεδομένα για βέλτιστη συνολική λειτουργία. Η χρήση μικροελεγκτών αποτελεί βασικό πυλώνα της σύγχρονης ηλεκτρονικής, με εφαρμογές που εκτείνονται από την καθημερινή τεχνολογία έως επιστημονικές συσκευές υψηλής ακρίβειας.

2.1.2 Single-Board Computer και Field-Programmable Gate Array

Μια άλλη κατηγορία ενσωματωμένων συστημάτων είναι οι **υπολογιστές μονού κυκλώματος** (SBC - Single-Board Computers). Πρόκειται για υπολογιστικά συστήματα που ενσωματώνουν όλα τα βασικά στοιχεία ενός υπολογιστή σε μία και μόνο πλακέτα κυκλωμάτων. Αποτελούνται από κεντρικό επεξεργαστή (CPU/GPU - Graphics Processing Unit), υπεύθυνο για την εκτέλεση εντολών (π.χ. ARM Cortex, Intel Atom), από μνήμη (RAM/ROM), είτε προσωρινή είτε μόνιμη, για την αποθήκευση δεδομένων, από εισόδους/εξόδους (I/O), όπως GPIO, USB, HDMI, Ethernet, Wi-Fi/Bluetooth, καθώς και από αποθηκευτικό χώρο, όπως eMMC, microSD ή SSD.

Ορισμένα βασικά χαρακτηριστικά των SBC είναι ο συμπαγής σχεδιασμός και το μικρό τους μέγεθος όσο μία πιστωτική κάρτα (π.χ. Raspberry Pi 4), η χαμηλή κατανάλωση ενέργειας (τροφοδοτούνται συνήθως με μπαταρίες ή μετασχηματιστές χαμηλής ισχύος, π.χ. 5V/2A), η πολυλειτουργικότητα (με υποστήριξη λειτουργικών συστημάτων όπως Linux, Android ή RTOS – Real Time OS), καθώς και το χαμηλό κόστος, το οποίο είναι σημαντικά μικρότερο σε σύγκριση με παραδοσιακούς υπολογιστές (π.χ. 35–150 €). Παραδείγματα υπολογιστών υλοποιημένων σε μία πλακέτα αποτελούν το Raspberry Pi 5 και το NVIDIA Jetson Orin Nano (βλπ Εικόνες 5 και 6).

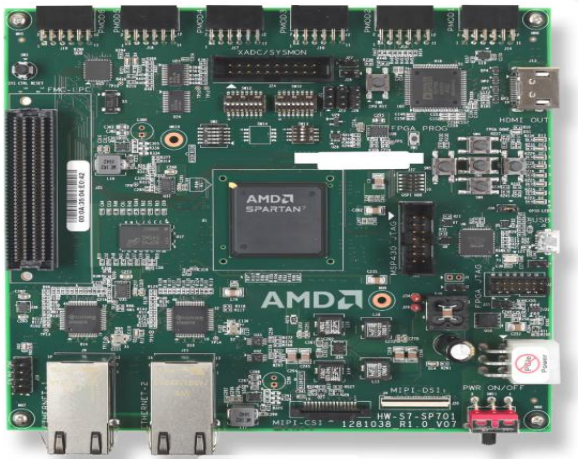


Εικόνα 5:Raspberry Pi 5
Πηγή:https://www.raspberrypi.com/products/raspberry-pi-5/

Εικόνα 6:NVIDIA Jetson Orin Nano
Πηγή:https://developer.nvidia.com/buy-jetson?product=all&location=GR

Μια επόμενη κατηγορία ενσωματωμένων συστημάτων είναι οι **διατάξεις προγραμματιζόμενων πυλών πεδίου** (FPGA – Field-Programmable Gate Array). Πρόκειται για έναν τύπο ολοκληρωμένου κυκλώματος που προσφέρει τη δυνατότητα σχεδιασμού και υλοποίησης παραμετροποιήσιμης λογικής. Αυτό επιτρέπει την ανάπτυξη πρωτοτύπων υψηλών επιδόσεων, τα οποία μπορούν να αποτελέσουν τη βάση για τον τελικό σχεδιασμό ενός συστήματος (βλπ. Εικόνες 7 και 8). Σε αντίθεση με άλλες κατηγορίες ολοκληρωμένων κυκλωμάτων, τα FPGA ξεχωρίζουν χάρη στη μεγάλη ευελιξία που προσφέρουν στον προγραμματισμό και την επαναπρογραμματισμό τους, καθώς επιτρέπουν την τροποποίηση της εσωτερικής τους αρχιτεκτονικής μέσω κατάλληλου λογισμικού. Αυτό το χαρακτηριστικό τα καθιστά ιδανικά για εφαρμογές που απαιτούν συνεχείς προσαρμογές, είτε λόγω μεταβαλλόμενων λειτουργικών αναγκών, είτε λόγω δύσκολων περιβαλλοντικών συνθηκών, όπως η χρήση σε εξωτερικούς χώρους. Επιπλέον, τα FPGA ανταποκρίνονται αποτελεσματικά στις διαρκώς αυξανόμενες απαιτήσεις της σύγχρονης τεχνολογίας, προσφέροντας μια ευέλικτη και δυναμική πλατφόρμα ανάπτυξης για καινοτόμες λύσεις.

Εικόνα 7:Altera Cyclone® V E FPGA Development Kit
Πηγή:https://www.intel.com/content/www/us/en/products/details/fpga/development-kits/cyclone/v-e.html



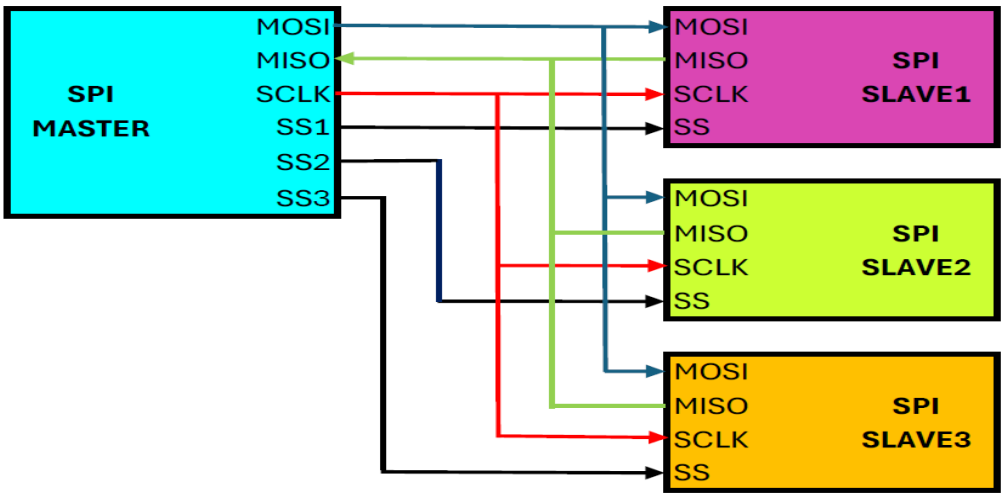
Εικόνα 8:AMD Spartan™ 7 SP701 FPGA Evaluation Kit
Πηγή:https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/

2.2 Ανάλυση πρωτοκόλλων επικοινωνίας του ESP32

Το ESP32 είναι ένας ισχυρός μικροελεγκτής, ο οποίος υποστηρίζει ποικιλία πρωτοκόλλων επικοινωνίας, τόσο ενσύρματων όσο και ασύρματων. Τα πρωτόκολλα αυτά επιτρέπουν την ανταλλαγή δεδομένων, τον έλεγχο αισθητήρων, τη σύνδεση με οθόνες, την επεξεργασία ήχου, καθώς και άλλες κρίσιμες λειτουργίες σε ενσωματωμένα συστήματα. Τρία από τα σημαντικότερα ενσύρματα πρωτόκολλα που υποστηρίζει το ESP32 είναι τα SPI [10] (Serial Peripheral Interface), I2S (Inter-IC Sound) και UART (Universal Asynchronous Receiver/Transmitter), καθένα από τα οποία εξειδικεύεται σε διαφορετικούς τομείς εφαρμογών. Όσον αφορά τα ασύρματα πρωτόκολλα, το βασικότερο είναι το Wi-Fi (IEEE 802.11 b/g/n), το οποίο επιτρέπει τη σύνδεση του μικροελεγκτή σε δίκτυα και την επικοινωνία με άλλες συσκευές σε πραγματικό χρόνο.

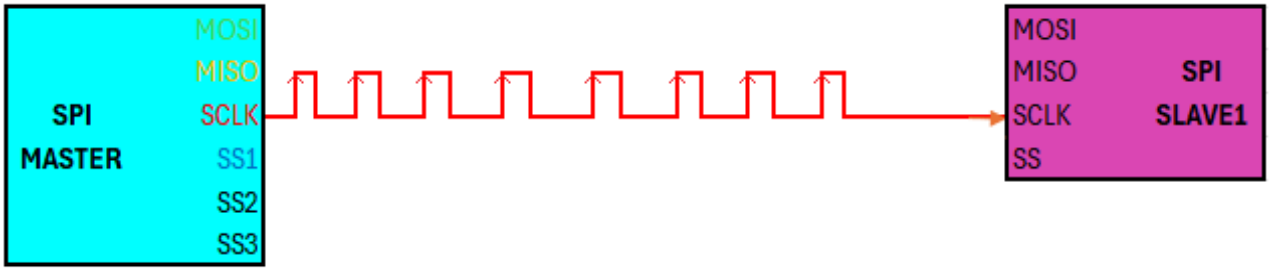
2.2.1 Πρωτόκολλο SPI

Ξεκινώντας πρώτα από τα ενσύρματα πρωτόκολλα θα αναλύσουμε πρώτα το SPI [10], το οποίο είναι μια σειριακή διεπαφή επικοινωνίας, η οποία υποστηρίζει πλήρους αμφίδρομη (full duplex) επικοινωνίας, δηλαδή μπορεί να στέλνει και να λαμβάνει δεδομένα ταυτόχρονα χάρη σε ξεχωριστές γραμμές δεδομένων για κάθε κατεύθυνση, τέτοιες γραμμές είναι η MISO (master in slave out) και η MOSI (master out slave in), η δε πρώτη είναι υπεύθυνη να μεταδίδει τα δεδομένα από το slave περιφερικό στον master μικροελεγκτή, ενώ η δεύτερη πραγματοποιεί το αντίστροφο, μεταδίδει τα δεδομένα από τον master μικροελεγκτή στον slave περιφερικό, τα δεδομένα αποστέλλονται με το πιο σημαντικό bit πρώτο (Most Significant Bit First - MSB First), δηλαδή για τον αριθμό 01000011 (δυαδικό), πρώτα αποστέλλεται το 0 (αριστερότερο bit), δεύτερο bit το 1, τρίτο bit το 0, κ.ο.κ (βλπ. Σχήμα 7). Βέβαια, για να επιτευχθεί συγχρονισμός χρησιμοποιείται ένα κοινό σήμα ρολογιού (SCK - Serial Clock), το γεγονός αυτό το καθιστά ένα σύγχρονο πρωτόκολλο, που είναι σχεδιασμένο για υψηλές ταχύτητες (συνήθως έως και 80MHz) και σύντομες αποστάσεις (εντός της ίδιας πλακέτας ή συστήματος). Το SPI χρησιμοποιείται ευρέως για τη διασύνδεση αποθηκευτικών μέσων (όπως κάρτες SD) και οθονών TFT. Τα βασικά του πλεονεκτήματα είναι η ταχύτητα, καθώς δεν χρειάζεται bit έναρξης και διακοπής, επιτρέποντας τη συνεχή μετάδοση δεδομένων, και η απλότητα, αφού δεν απαιτείται περίπλοκο σύστημα διευθυνσιοδότησης slave συσκευών. Ως μειονεκτήματα θεωρούνται η έλλειψη επιβεβαίωσης λήψης των δεδομένων και το γεγονός ότι απαιτούνται τέσσερα καλώδια ανά slave συσκευή (βλ. Σχήμα 3). Στην περίπτωση ελέγχου πολλών slave συσκευών, απαιτείται αντίστοιχος αριθμός θυρών Slave Select (SS1, SS2, SS3, ...), ώστε ο master μικροελεγκτής να μπορεί να επιλέγει ποια slave συσκευή θα ενεργοποιεί.

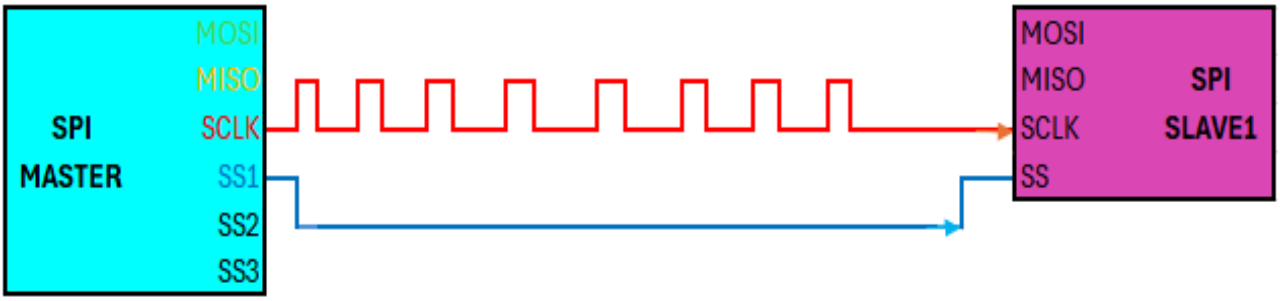


Σχήμα 3:Συνδεσμολογία πρωτοκόλλου SPI

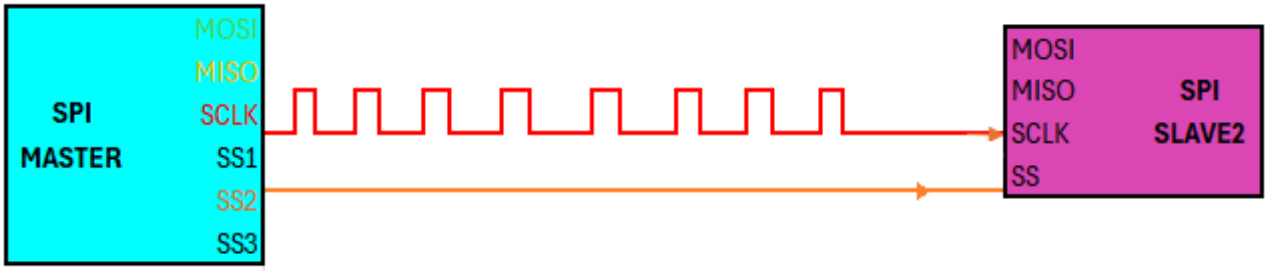
Η ταχύτητα και η απλότητα του πρωτοκόλλου SPI [10] βασίζονται άμεσα στην αρχή λειτουργίας του, η οποία περιγράφεται παρακάτω. Συγκεκριμένα, ένα κοινό σήμα ρολογιού συγχρονίζει τη μεταφορά δεδομένων μεταξύ ενός master και ενός ή περισσότερων slave συσκευών, τόσο για αποστολή όσο και για λήψη. Κάθε παλμός του ρολογιού αντιστοιχεί στη μετάδοση ενός bit, ενώ η συχνότητα του ρολογιού καθορίζει την ταχύτητα επικοινωνίας (π.χ., συχνότητα 1 MHz → ταχύτητα 1 Mbps). Η πολικότητα και η φάση του ρολογιού έχουν οριστεί ώστε η κατάσταση ηρεμίας να είναι χαμηλή (Low, 0 Volt) και τα δεδομένα να δειγματοληπτούνται στην ανερχόμενη ακμή του παλμού (Rising edge), όπως φαίνεται στο Σχήμα 4. Το σήμα ρολογιού δημιουργείται από τη master συσκευή και χρησιμοποιείται από τις slave συσκευές για να συγχρονίσουν τη λήψη των δεδομένων. Η επιλογή της slave συσκευής που θα λάβει τα δεδομένα γίνεται από τη master συσκευή, η οποία θέτει τη θύρα SS (Slave Select) σε χαμηλή τάση (0 Volt), ενεργοποιώντας έτσι την αντίστοιχη slave συσκευή (βλ. Σχήμα 5). Οι υπόλοιπες slave συσκευές παραμένουν ανενεργές, καθώς η θύρα SS τους διατηρείται σε υψηλή τάση (3,3 Volt) (βλ. Σχήμα 6). Στο επόμενο βήμα, η master συσκευή στέλνει τα δεδομένα ένα bit τη φορά προς τη slave συσκευή μέσω της γραμμής MOSI (master out slave in) (βλ. Σχήμα 7). Εφόσον απαιτείται απόκριση, η slave συσκευή μπορεί να επιστρέψει δεδομένα (π.χ. τον δυαδικό αριθμό 11100011) ένα bit κάθε φορά, μέσω της γραμμής MISO (master in slave out), προς τη master συσκευή (βλ. Σχήμα 8).



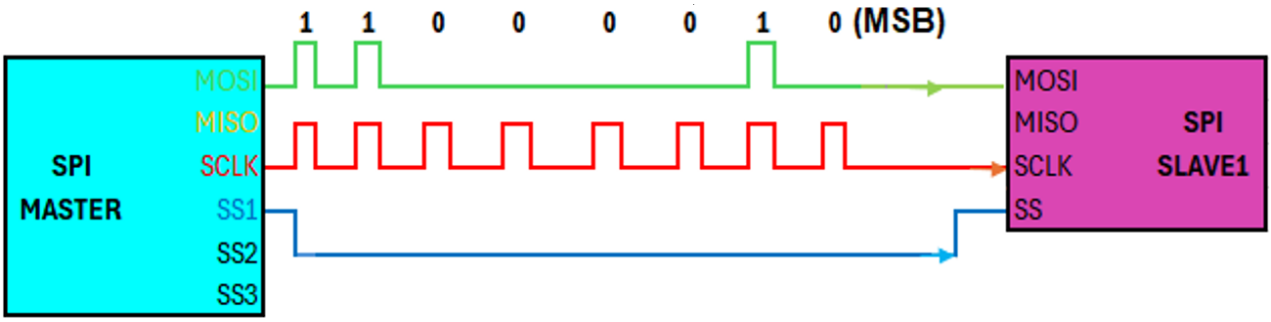
Σχήμα 4:Η master συσκευή παράγει το σήμα ρολογιού.



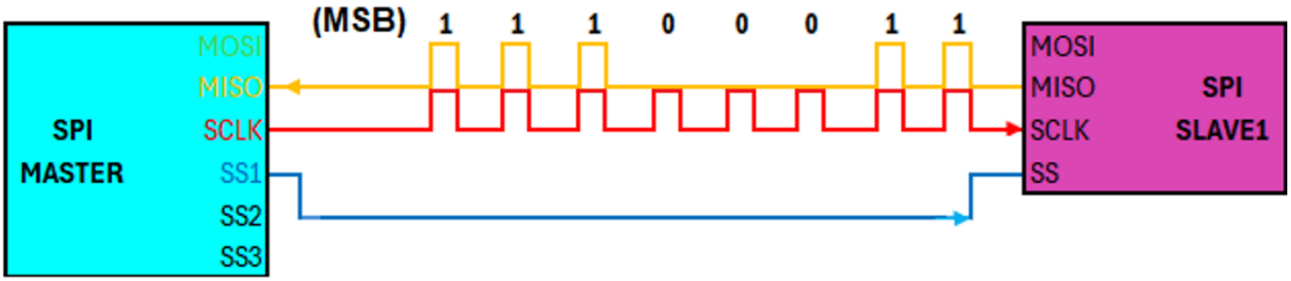
Σχήμα 5:Ενεργοποίηση της slave συσκευής.



Σχήμα 6:Απενεργοποίηση την slave συσκευής

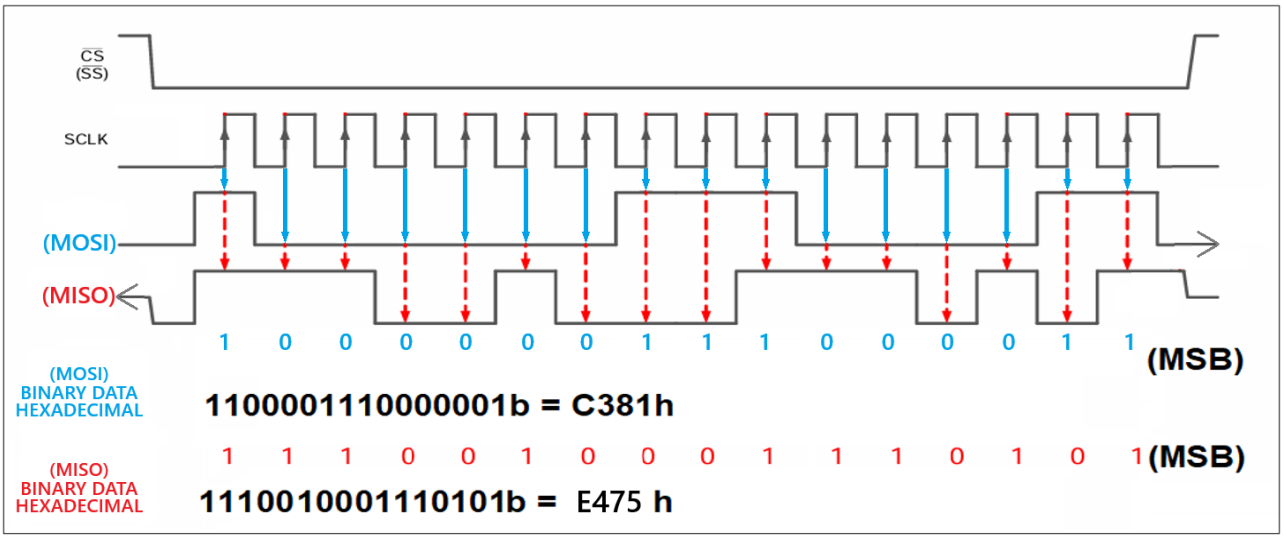


Σχήμα 7:Αποστολή διαδοχικών bit κατά μήκος της γραμμής MOSI.



Σχήμα 8:Αποστολή διαδοχικών bit κατά μήκος της γραμμής MISO.

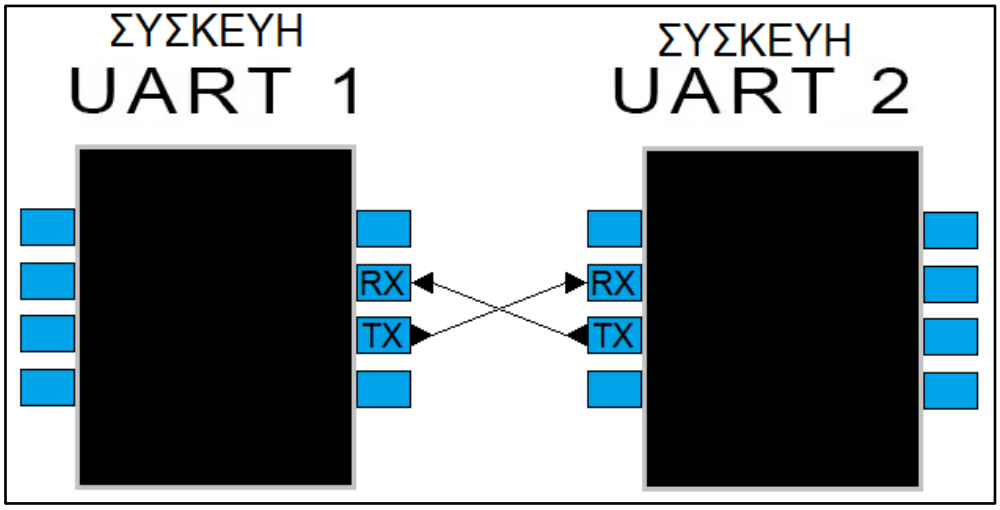
Παρακάτω απεικονίζεται ένα αναλυτικό παράδειγμα πλήρους αμφίδρομης επικοινωνίας (βλπ. Σχήμα 9).



Σχήμα 9:Παράδειγμα πλήρους αμφίδρομη επικοινωνίας

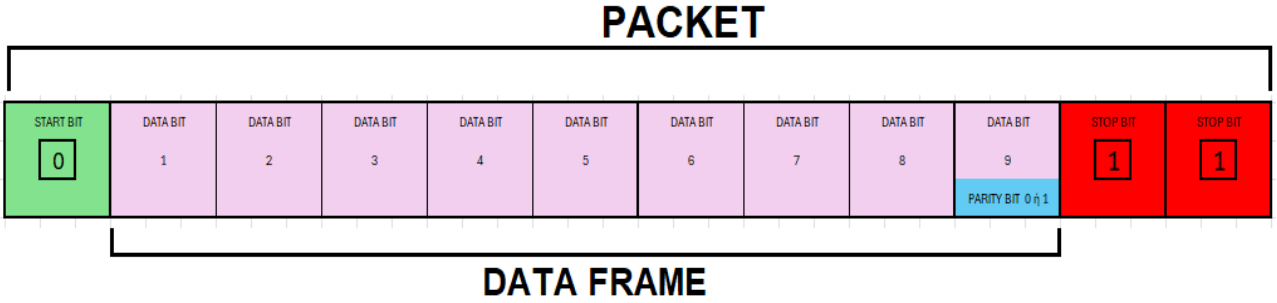
2.2.2 Πρωτόκολλο UART

Ένα εξίσου σημαντικό πρωτόκολλο επικοινωνίας είναι το UART (Universal Asynchronous Receiver/Transmitter) [11]. Πρόκειται για ένα ασύγχρονο πρωτόκολλο που χρησιμοποιείται για τη σειριακή μετάδοση δεδομένων μεταξύ δύο συσκευών. Δεν απαιτεί κοινό σήμα ρολογιού (Clock), αλλά βασίζεται σε συγκατάθεση ρυθμού μετάδοσης (Baud rate), οι συσκευές πρέπει να έχουν ίδιο baud rate (bits ανά δευτερόλεπτο) και bits συγχρονισμού (start/stop bits) για την ορθή ανταλλαγή πληροφοριών. Οι δυο συσκευές UART επικοινωνούν απευθείας μεταξύ τους, χρησιμοποιώντας μόνο δύο καλώδια για τη μετάδοση δεδομένων. Τα δεδομένα κατευθύνονται από τον ακροδέκτη TX του UART που εκπέμπει προς τον ακροδέκτη RX του UART που λαμβάνει (βλπ. Σχήμα 10).



Σχήμα 10:Συνδεσμολογία πρωτοκόλλου UART

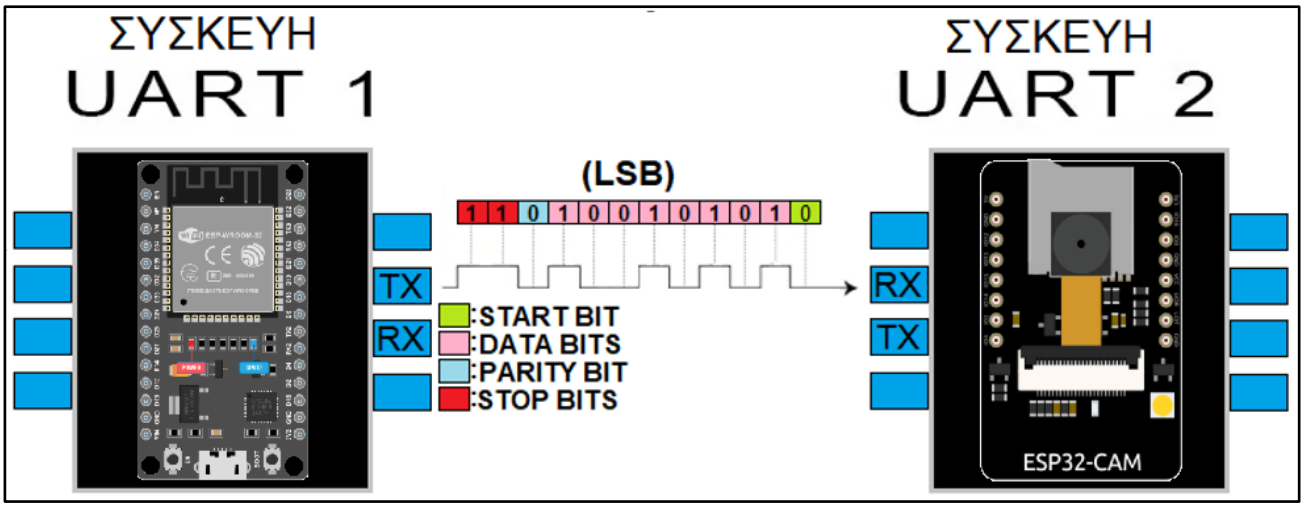
Η αρχή λειτουργίας του πρωτοκόλλου βασίζεται σε μια συγκεκριμένη δομή πακέτου δεδομένων, η οποία αποτελείται από το **Start Bit** (1 bit), το οποίο ενεργοποιεί τη μετάδοση όταν τεθεί στη λογική κατάσταση 0 (bit = 0), στη συνέχεια ακολουθούν τα **Data Bits** (5 έως 9 bits), όπου μεταφέρεται το πραγματικό μήνυμα (συνήθως 8 bits), έπειτα τοποθετείται προαιρετικά το **Parity Bit** (1 bit), το οποίο χρησιμοποιείται για τον έλεγχο σφαλμάτων και τέλος προστίθενται τα **Stop Bit(s)** (1 ή 2 bits), τα οποία δηλώνουν τον τερματισμό του πακέτου, όταν τεθούν στη λογική κατάσταση 1 (bit = 1) (βλ. Σχήμα 11).



Σχήμα 11:Δομή πακέτου δεδομένων

Το πρωτόκολλο UART (Universal Asynchronous Receiver/Transmitter) χρησιμοποιείται ευρέως για ασύγχρονη σειριακή επικοινωνία μεταξύ δύο συσκευών. Κατά τη μετάδοση δεδομένων, το UART [10] του πομπού προσθέτει ειδικά bits έναρξης και διακοπής στο πακέτο δεδομένων. Τα bits αυτά χρησιμεύουν στον καθορισμό των ορίων του πακέτου, επιτρέποντας στο UART του δέκτη να αναγνωρίσει την έναρξη και το τέλος της μετάδοσης. Όταν ο δέκτης εντοπίσει το bit έναρξης, ξεκινά την ανάγνωση των εισερχόμενων bit, ξεκινώντας από το λιγότερο σημαντικό bit (Least Significant Bit First - LSB First), με ρυθμό προκαθορισμένης συχνότητας, γνωστό ως ρυθμός baud. Ο ρυθμός baud αποτελεί μέτρο της ταχύτητας μετάδοσης δεδομένων και εκφράζεται σε bit ανά δευτερόλεπτο (bps). Για να επιτευχθεί επιτυχής επικοινωνία, και οι δύο συσκευές πρέπει να είναι ρυθμισμένες στον ίδιο ρυθμό baud. Η γραμμή μετάδοσης UART διατηρείται σε υψηλή λογική στάθμη (λογικό 1) όταν δεν πραγματοποιείται μεταφορά δεδομένων. Η έναρξη της μετάδοσης σηματοδοτείται από μια μετάβαση της γραμμής σε χαμηλή στάθμη (λογικό 0) για τη διάρκεια ενός κύκλου ρολογιού, η οποία αναγνωρίζεται από τον δέκτη ως ένδειξη έναρξης. Το πλαίσιο δεδομένων περιλαμβάνει τα πραγματικά bit πληροφορίας που μεταφέρονται, και μπορεί να αποτελείται από 5 έως 8 bit. Εφόσον δεν χρησιμοποιείται bit ισοτιμίας (parity bit), είναι δυνατή και η χρήση πλαισίου 9 bit. Το bit ισοτιμίας χρησιμοποιείται για τη βασική ανίχνευση σφαλμάτων κατά τη μετάδοση. Ο δέκτης, αφού αναγνώσει τα δεδομένα, υπολογίζει το πλήθος των bit με τιμή 1 και το συγκρίνει με το bit ισοτιμίας. Στην περίπτωση ζυγής ισοτιμίας, το συνολικό πλήθος των bit με τιμή 1 (μαζί με το bit ισοτιμίας) πρέπει να είναι άρτιος αριθμός, ενώ στην περίπτωση περιττής ισοτιμίας,

πρέπει να είναι περιττός. Αν η συνθήκη αυτή δεν ικανοποιείται, τότε ο δέκτης αντιλαμβάνεται ότι έχει σημειωθεί σφάλμα κατά τη μετάδοση. Η λήξη της μετάδοσης δηλώνεται από τον πομπό με την επαναφορά της γραμμής μετάδοσης σε υψηλή στάθμη (λογικό 1), η οποία διατηρείται για τουλάχιστον δύο κύκλους ρολογιού. Παρακάτω αναλύεται ένα παράδειγμα αποστολής ενός 8 bits (10010101b) μήνυμα (βλπ. Σχήμα 12).

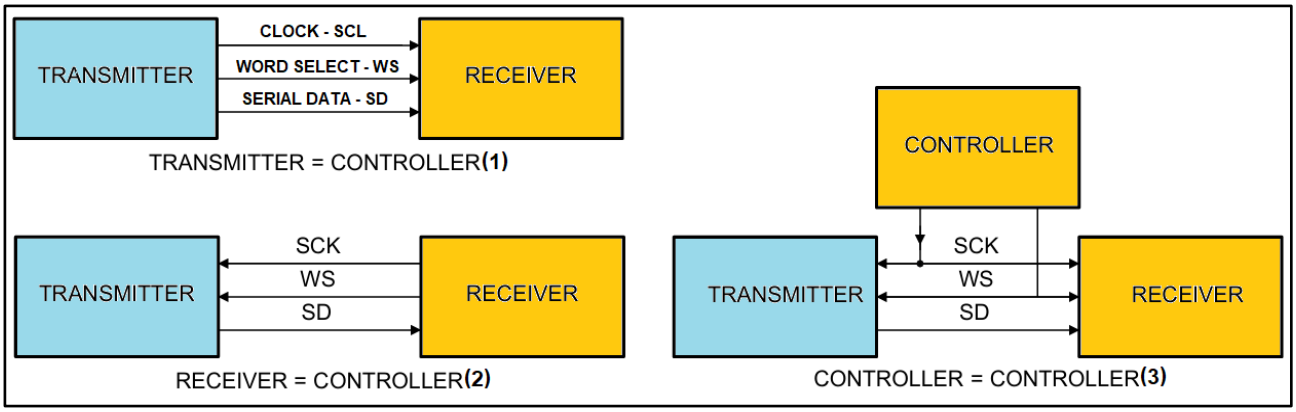


Σχήμα 12:Παράδειγμα επικοινωνίας UART

2.2.3 Πρωτόκολλο I2S

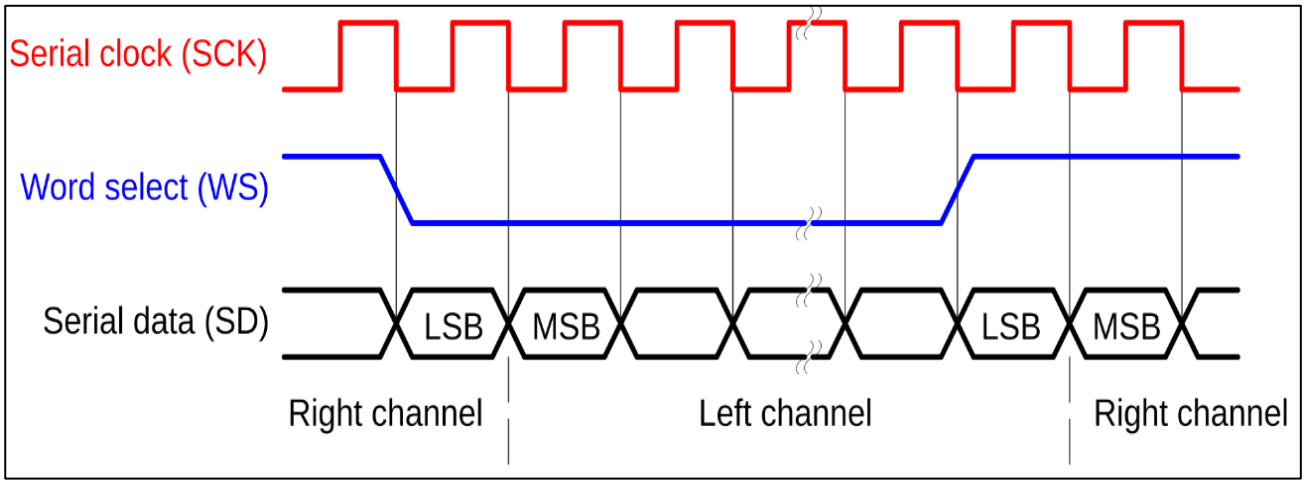
Το πρωτόκολλο Inter-Integrated Circuit Sound (I^2S ή I2S) αποτελεί ένα διαδεδομένο πρότυπο σειριακής επικοινωνίας, το οποίο χρησιμοποιείται για τη μετάδοση ψηφιακού ήχου ανάμεσα σε ολοκληρωμένα κυκλώματα εντός ηλεκτρονικών συστημάτων [12]. Αναπτύχθηκε αρχικά από την εταιρεία Philips και προσφέρει έναν απλό και αποδοτικό τρόπο μεταφοράς στερεοφωνικού ήχου σε μορφή διαμόρφωσης παλμικού κώδικα (PCM - Pulse Code Modulation). Σε αντίθεση με ασύγχρονα πρωτόκολλα, το I^2S διαχωρίζει τα σήματα του ρολογιού και των δεδομένων, γεγονός που διευκολύνει τον σχεδιασμό του δέκτη και ενισχύει τον συγχρονισμό και την αξιοπιστία της μετάδοσης. Ο δίαυλος του I^2S περιλαμβάνει συνήθως τρεις βασικές γραμμές: μία για τα σειριακά δεδομένα ήχου (SD - Serial data), μία για την επιλογή λέξης (WS - Word select) και μία για το ρολόι bit (Clock), η συνδεσμολογία αυτών των γραμμών, μπορεί να διαμορφωθεί με τρεις διαφορετικούς τρόπους, ανάλογα με ποια συσκευή (Πομπός ή Δέκτης) ελέγχει την επιλογή λέξης (γνωστή και ως ρολόι καναλιού ή αριστερά-δεξιά) και το ρολόι bit.

Σε πολύπλοκα συστήματα, ωστόσο, μπορεί να υπάρχουν αρκετοί πομποί και δέκτες, γεγονός που καθιστά δύσκολο τον ορισμό του ελεγκτή. Σε τέτοια συστήματα, ένας ελεγκτής συστήματος ελέγχει τη ροή δεδομένων ψηφιακού ήχου μεταξύ των συσκευών, συγκεκριμένα ο ελεγκτής μπορεί να συνδυαστεί με πομπούς ή δέκτες και να ελέγχει την ενεργοποίηση/απενεργοποίηση τους, είτε μέσω λογισμικού είτε με προγραμματισμό των ακροδεκτών (pins) τους. Παρακάτω παρουσιάζονται οι τρεις συνδεσμολογίες του πρωτοκόλλου (βλπ. Σχήμα 13). Η δομή αυτή επιτρέπει τον ακριβή συγχρονισμό κατά την αποστολή του ήχου, καθιστώντας το ιδιαίτερα κατάλληλο για εφαρμογές υψηλής πιστότητας, όπως μετατροπείς ψηφιακού σε αναλογικό (DAC), αναλογικού σε ψηφιακό (ADC), συστήματα επεξεργασίας ήχου και μικροελεγκτές.



Σχήμα 13:Τρόποι συνδεσμολογία συσκευών I2S

Η αρχή λειτουργίας του πρωτοκόλλου είναι σχετικά απλή και βασίζεται στην συγχρονισμένη μετάδοση δεδομένων ήχου (βλπ. Σχήμα 14). Όπως αναφέραμε απαιτούνται μόνο 3 βασικές γραμμές για λειτουργία του, αρχικά χρειαζόμαστε, τα δεδομένα ήχου(SD), τα οποία αποστέλλονται με το πιο σημαντικό bit (MSB) να μεταδίδεται πρώτο, δεύτερον την γραμμή Word select (WS) για να καθορίζουμε ποιο κανάλι (αριστερό ή δεξί) μεταδίδεται, θέτοντας το WS = 0 ενεργοποιούμε το αριστερό κανάλι, ενώ με WS = 1 ενεργοποιούμε το δεξί κανάλι ήχου. Η αλλαγή των καναλιών πραγματοποιείται είτε στην ανιούσα (Leading edge) είτε στην κατιούσα ακμή(Trailing edge) του σειριακού ρολογιού, προεπιλεγμένη είναι ανιούσα ακμή. Η μετάβαση από ανά κανάλι στο άλλο απαιτεί μία περίοδο ρολογιού πριν μεταδοθεί το επόμενο MSB. Αυτό επιτρέπει στον πομπό να συγχρονίσει τη μετάδοση του σειριακού δεδομένου, το οποίο θα ρυθμιστεί κατάλληλα πριν ξεκινήσει η μετάδοση. Επιπλέον, δίνει τη δυνατότητα στον δέκτη να αποθηκεύσει την προηγούμενη λέξη (σύνολο bit) και να καθαρίσει την είσοδο για την επόμενη λέξη. Και τέλος ένα ρολόι που να παρέχει τον ρυθμό μετάδοσης των bits, η συχνότητα του ρολογιού εξαρτάτε από την συχνότητα και ανάλυση της δειγματοληψίας του ήχου καθώς και από τον αριθμό των καναλιών. Για παράδειγμα αν αποστέλλουμε δεδομένα ήχου με ποιότητα CD, τότε χρειάζεται να ρυθμιστεί το ρολόι στα 1.4112 MHz (44.1 kHz × 16 bits × 2 κανάλια = 1.4112 MHz).



Σχήμα 14:Διάγραμμα χρονισμού I2S
Πηγή:https://commons.wikimedia.org/w/index.php?curid=16579640

2.2.4 Πρωτόκολλο Wi-Fi (IEEE 802.11)

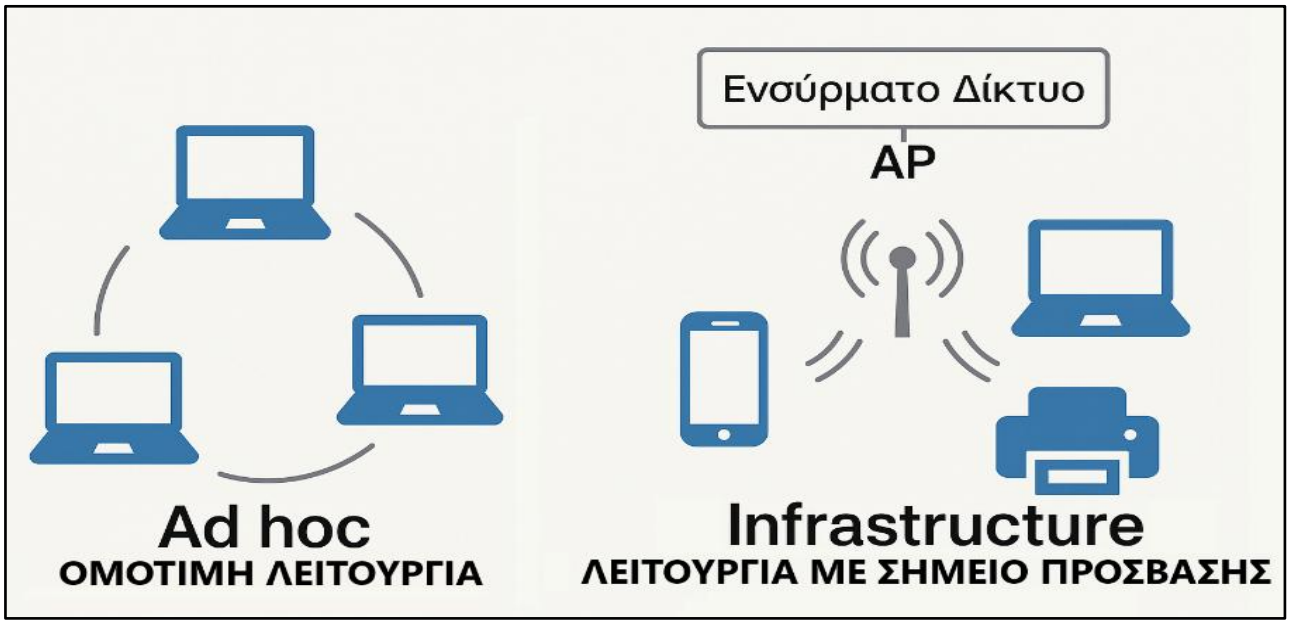
Το Wi-Fi (Wireless Fidelity) αποτελεί μια διαδεδομένη τεχνολογία ασύρματης δικτύωσης, η οποία δίνει τη δυνατότητα σε συσκευές όπως έξυπνα κινητά τηλέφωνα, φορητούς υπολογιστές, ταμπλέτες και IoT συσκευές να συνδέονται είτε στο διαδίκτυο, είτε μεταξύ τους, μέσω ενός τοπικού ασύρματου δικτύου (WLAN) [13]. Η τεχνολογία Wi-Fi βασίζεται στη σειρά

προτύπων IEEE 802.11, τα οποία έχουν εξελιχθεί σημαντικά με την πάροδο του χρόνου, τόσο από άποψη ταχύτητας όσο και λειτουργικότητας. Η αρχική χρήση του όρου Wi-Fi αφορούσε συσκευές συμβατές με τα πρότυπα IEEE 802.11b/g/n, που λειτουργούν στη συχνότητα των 2.4 GHz. Ωστόσο, σήμερα ο όρος χρησιμοποιείται γενικά για να περιγράψει κάθε τύπο ασύρματης τοπικής δικτύωσης ανεξαρτήτως πρότυπου ή φάσματος συχνοτήτων. Ο παρακάτω πίνακας (βλπ. Πίνακας 2) παρουσιάζει τις εκδόσεις Wi-Fi, τα πρότυπα IEEE, τις χρονολογίες κυκλοφορίας, τη μέγιστη ταχύτητα δεδομένων και τη ζώνη συχνοτήτων λειτουργίας κάθε έκδοσης. Βέβαια, ο ρυθμός μετάδοσης δεδομένων στο 802.11 εξαρτάται από την απόσταση μεταξύ των κόμβων. Όσο πιο μακριά βρίσκεται η ασύρματη συσκευή από το σημείο πρόσβασης, τόσο χαμηλότερη είναι η ταχύτητα.

Πίνακας 2:Εξέλιξη προτύπου IEEE 802.11

Έκδοση	Πρότυπο IEEE	Έτος	Μέγιστος ρυθμός δεδομένων (Mbit/s)	Συχνότητα (GHz)
Wi-Fi 7	802.11be	2024	1376 έως 46120	2.4/5/6
Wi-Fi 6E	802.11ax	2020	574 έως 9608	6
Wi-Fi 6	802.11ax	2019	574 έως 9608	2.4/5
Wi-Fi 5	802.11ac	2014	433 έως 6933	5
Wi-Fi 4	802.11n	2008	72 έως 600	2.4/5
(Wi-Fi 3)	802.11g	2003	6 έως 54	2.4
(Wi-Fi 2)	802.11a	1999	6 έως 54	5
(Wi-Fi 1)	802.11b	1999	1 έως 11	2.4
(Wi-Fi 0)	802.11	1997	1 έως 2	2.4

Το πρωτόκολλο IEEE 802.11 [12] υποστηρίζει δύο βασικά λειτουργικά μοντέλα, το Ad hoc (ομότιμη λειτουργία) και το Infrastructure mode (λειτουργία με σημείο πρόσβασης). Στο πρώτο μοντέλο οι συσκευές συνδέονται άμεσα μεταξύ τους, χωρίς την ύπαρξη κάποιου κεντρικού κόμβου, όλοι οι κόμβοι είναι ισότιμοι (peer-to-peer) και η πρόσβαση στο κοινό ασύρματο μέσο (ραδιοσυχνοτικό φάσμα) ρυθμίζεται μέσω αλγορίθμων κατανομής πρόσβασης. Στο δεύτερο μοντέλο λειτουργείας υπάρχει ένας κεντρικός κόμβος, το σημείο πρόσβασης (access point), που συνήθως συνδέεται με το ενσύρματο δίκτυο (π.χ. ethernet, Internet) και το οποίο αναλαμβάνει τον έλεγχο πρόσβασης στο δίκτυο, λειτουργεί και ως αμφίδρομος επαναλήπτης (repeater) και διαχειρίζεται τη σύνδεση των συσκευών πελατών (βλπ. Εικόνα 9).



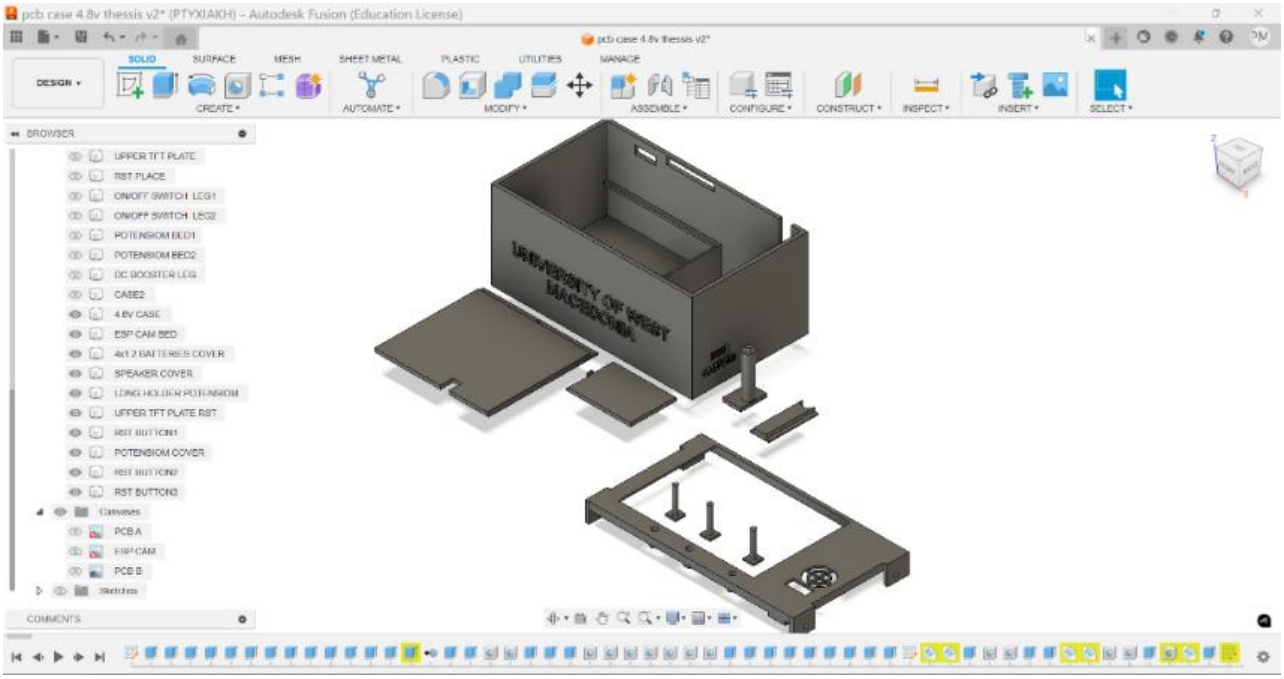
Εικόνα 9:Βασικά λειτουργικά μοντέλα πρωτοκόλλου IEEE 802.11

2.3 Λογισμικά

Για την διεκπεραίωση - υλοποίηση της διπλωματικής εργασίας, δημιουργήθηκε η ανάγκη για τη χρήση σύγχρονων λογισμικών και εργαλείων που υποστηρίζουν τον σχεδιασμό, την κατασκευή και τη λειτουργία ενός ολοκληρωμένου συστήματος, ονομαστικά τα λογισμικά που χρησιμοποιήθηκαν είναι, το Autodesk Fusion 360, το EasyEDA, το Creality Print, η πλατφόρμα Microsoft Azure, και τέλος, το Arduino IDE. Ο συνδυασμός αυτών των τεχνολογιών εξασφαλίζει μια ολοκληρωμένη και καινοτόμα προσέγγιση στην ανάπτυξη του έργου.

2.3.1 Λογισμικό Autodesk Fusion 360

Το Fusion 360 ανήκει στην εταιρεία Autodesk [14]. Η Autodesk είναι μια αμερικανική πολυεθνική εταιρεία που ειδικεύεται στην ανάπτυξη λογισμικού για σχεδίαση, μηχανική, αρχιτεκτονική και κατασκευές. Το Fusion 360 αποτελεί ένα ολοκληρωμένο λογισμικό τρισδιάστατου σχεδιασμού (Computer-Aided Design), μηχανικής ανάλυσης (Computer-Aided Engineering) και κατεργασιών (Computer-Aided Manufacturing), που χρησιμοποιείται ευρέως στον χώρο της μηχανολογίας και της βιομηχανικής σχεδίασης. Η ενσωμάτωση όλων αυτών των λειτουργιών σε μία ενιαία πλατφόρμα δίνει τη δυνατότητα για πιο αποδοτική ροή εργασίας, από τη σύλληψη της ιδέας μέχρι την τελική παραγωγή. Στο πλαίσιο της διπλωματικής μου εργασίας, το Fusion 360 χρησιμοποιήθηκε για τον σχεδιασμό εξαρτημάτων με μεγάλη ακρίβεια, προσφέροντας δυνατότητες παραμετρικής σχεδίασης, καθώς και εξαγωγής αρχείων κατάλληλων για 3D εκτύπωση. Επιπλέον, η δυνατότητα συνεργασίας στο cloud επιτρέπει την εύκολη πρόσβαση και τροποποίηση αρχείων. Η χρήση του Fusion 360 συνέβαλε καθοριστικά στην επίτευξη υψηλής ποιότητας σχεδιαστικών αποτελεσμάτων, μειώνοντας τα λάθη και τον χρόνο παραγωγής. Στην παρακάτω εικόνα παρουσιάζεται το περιβάλλον εργασίας του λογισμικού (βλπ. Εικόνα 10).

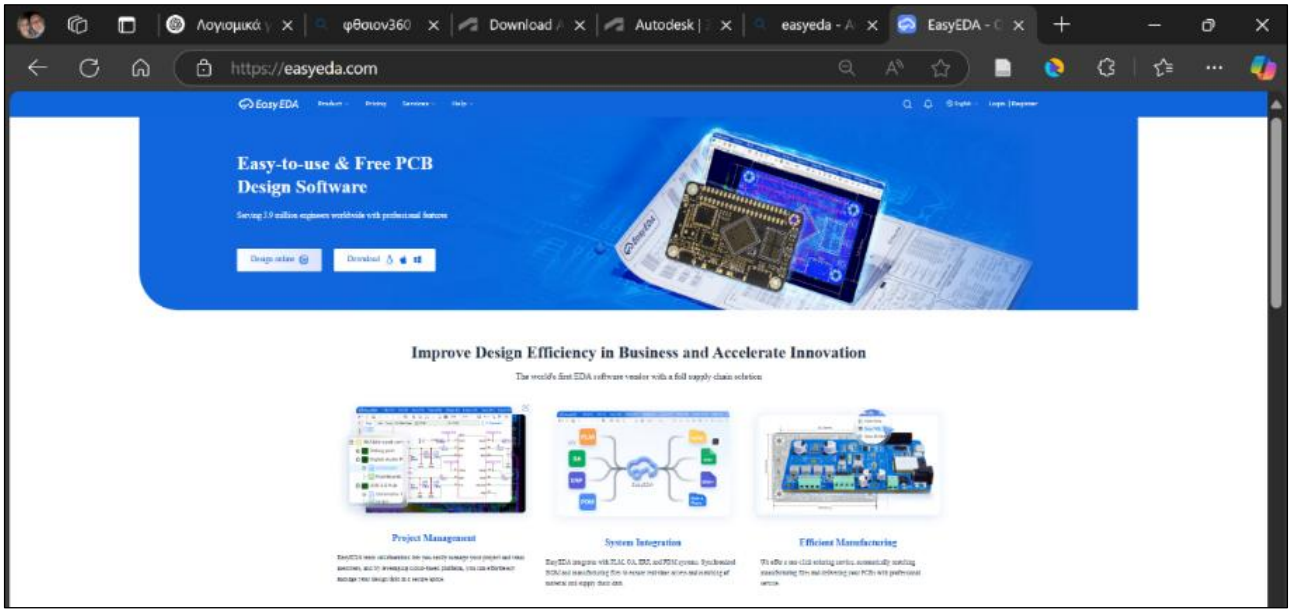


Εικόνα 10:Περιβάλλον 3D σχεδιασμού του Fusion360

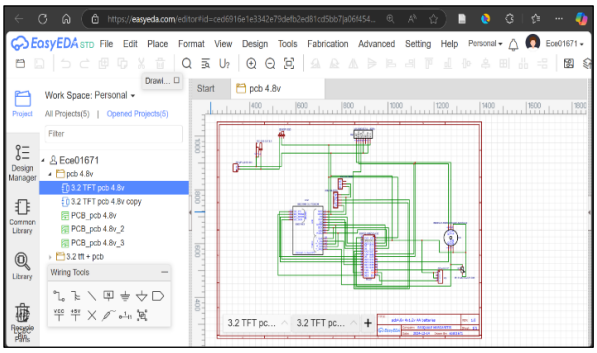
2.3.2 Λογισμικό EasyEDA

Το EasyEDA είναι ένα διαδικτυακό εργαλείο σχεδίασης ηλεκτρονικών κυκλωμάτων και πλακετών τυπωμένων κυκλωμάτων (PCB), το οποίο ιδρύθηκε το 2010 από τους Dillon He και Eric Cui [15][16]. Από τότε, έχει εξελιχθεί σε σημαντικό τμήμα του JLC Group, μιας κινεζικής

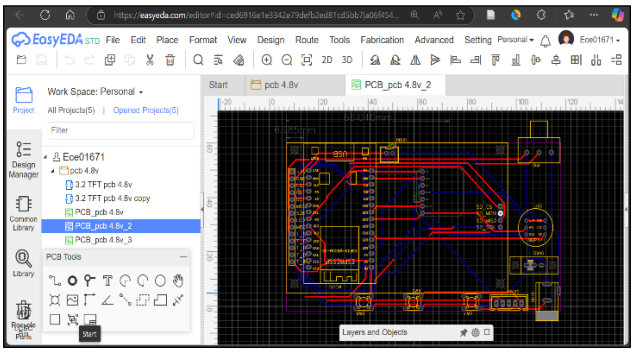
εταιρείας, η οποία περιλαμβάνει επίσης τις JLCPCB και LCSC, προσφέροντας ολοκληρωμένες υπηρεσίες σχεδίασης και κατασκευής ηλεκτρονικών κυκλωμάτων. Η πλατφόρμα EasyEDA επιτρέπει στους χρήστες να σχεδιάζουν, να προσομοιώνουν και να κατασκευάζουν ηλεκτρονικά κυκλώματα μέσω ενός ενιαίου, φιλικού προς τον χρήστη περιβάλλοντος, ενσωματώνοντας εργαλεία όπως SPICE (Simulation Program with Integrated Circuit Emphasis) προσομοίωση, σχεδίαση PCB και άμεση σύνδεση με υπηρεσίες κατασκευής. Παρακάτω απεικονίζεται η αρχική σελίδα του λογισμικού (βλπ. Εικόνα 11) καθώς και το περιβάλλον εργασίας (βλπ. Εικόνες 12,13).



Εικόνα 11:Αρχική σελίδα λογισμικού EasyEDA
Πηγή:https://easyeda.com/



Εικόνα 12:Περιβάλλον σχεδιασμού ηλεκτρονικού σχεδίου EasyEDA
Πηγή: EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool

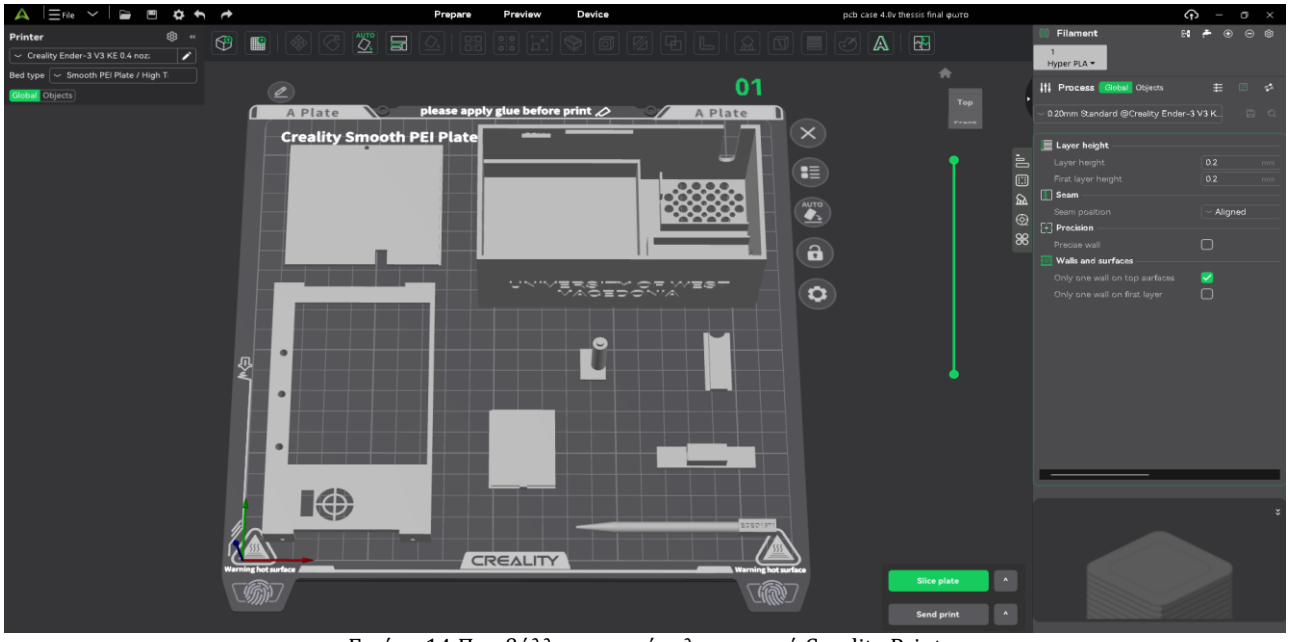


Εικόνα 13:Περιβάλλον σχεδιασμού ηλεκτρονικής πλακέτας EasyEDA
Πηγή: EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool

2.3.3 Λογισμικό Creality Print

Το Creality Print είναι ένα εξειδικευμένο λογισμικό τρισδιάστατης εκτύπωσης που έχει αναπτυχθεί από την εταιρεία Creality, μία από τις πιο γνωστές κατασκευάστριες 3D εκτυπωτών παγκοσμίως [17]. Το πρόγραμμα λειτουργεί ως slicer, δηλαδή μετατρέπει τα τρισδιάστατα μοντέλα (.STL, .OBJ κ.ά.) σε αρχεία G-code, τα οποία μπορούν να διαβαστούν και να εκτυπωθούν από 3D εκτυπωτές της Creality. Με ένα φιλικό και εύχρηστο περιβάλλον, το Creality Print προσφέρει μια πληθώρα ρυθμίσεων για τον έλεγχο παραμέτρων εκτύπωσης όπως η ταχύτητα, η θερμοκρασία, το ύψος στρώσης και η πλήρωση (infill). Παράλληλα, διαθέτει δυνατότητες προεπισκόπησης της εκτύπωσης, υποστήριξη για πολλαπλά μοντέλα και

αυτόματη δημιουργία στηρίξεων (supports). Το Creality Print διευκολύνει την προετοιμασία εκτυπώσεων με ακρίβεια και αξιοπιστία, αποτελώντας απαραίτητο εργαλείο για χρήστες 3D εκτυπωτών της συγκεκριμένης εταιρείας. Το Creality Print συμβάλλει στην προετοιμασία και τη βελτιστοποίηση των αρχείων για 3D εκτύπωση, καθιστώντας δυνατή την υλοποίηση των φυσικών μοντέλων. Παρακάτω απεικονίζεται το περιβάλλον εργασίας του λογισμικού (βλπ. Εικόνα 14).



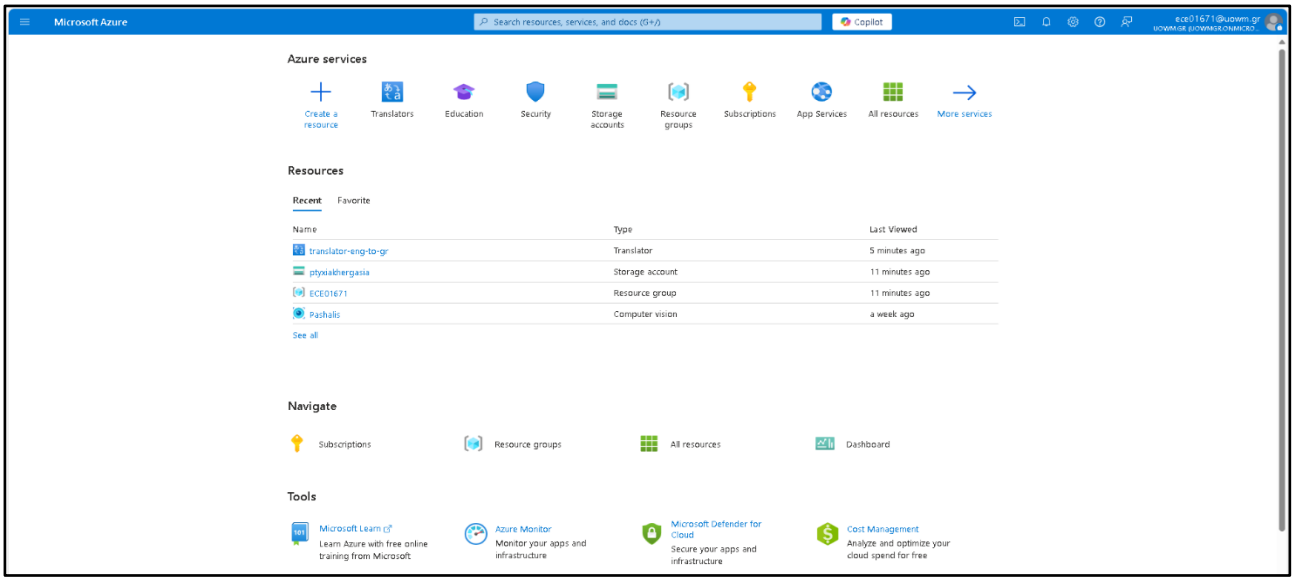
Εικόνα 14:Περιβάλλον εργασίας λογισμικού Creality Print

2.3.4 Λογισμικό Microsoft Azure

Το Microsoft Azure αποτελεί μια ολοκληρωμένη πλατφόρμα υπηρεσιών cloud [18], η οποία παρέχει ένα ευρύ φάσμα εργαλείων για την ανάπτυξη, φιλοξενία και διαχείριση εφαρμογών και έξυπνων συστημάτων. Στο πλαίσιο της παρούσας διπλωματικής εργασίας, το Azure αξιοποιείται κυρίως για τις δυνατότητές του στην υπολογιστική όραση (Computer Vision) και τη διαχείριση φωτογραφιών μέσω containerized εφαρμογών. Η πλατφόρμα προσφέρει υψηλή επεκτασιμότητα και διαθεσιμότητα, επιτρέποντας την αξιοποίηση ισχυρών υπολογιστικών πόρων μέσω του cloud, χωρίς την ανάγκη ισχυρού τοπικού εξοπλισμού.

Οι υπηρεσίες τεχνητής νοημοσύνης του Azure, και ειδικότερα το **Azure computer vision API**, δίνουν τη δυνατότητα ανάλυσης εικόνων και εξαγωγής μεταδεδομένων σε πραγματικό χρόνο, κάτι που είναι κρίσιμο για λειτουργίες έξυπνης οπτικής αναγνώρισης [19]. Παράλληλα, το **Azure blob storage** χρησιμοποιείται για την ασφαλή και αποδοτική αποθήκευση μεγάλων όγκων μη δομημένων δεδομένων, όπως φωτογραφιών, βίντεο και αρχείων εφαρμογών. Επιτρέπει την αποθήκευση, πρόσβαση και διαχείριση αυτών των δεδομένων με χαμηλό κόστος και υψηλή διαθεσιμότητα, υποστηρίζοντας τη συνεχή ροή δεδομένων μεταξύ των ενσωματωμένων συσκευών και του cloud [20].

Επιπλέον, ενσωματώνεται το **Azure translator**, μία υπηρεσία αυτόματης μετάφρασης που βασίζεται σε τεχνητή νοημοσύνη, με σκοπό τη μετάφραση της απάντησης του Azure Computer Vision API από τα Αγγλικά στα Ελληνικά. Με αυτόν τον τρόπο, η μετάφραση των αποτελεσμάτων ανάλυσης εικόνας επιτρέπει στον τελικό χρήστη να τα κατανοεί άμεσα, διευκολύνοντας τη χρήση του συστήματος σε ελληνικό γλωσσικό περιβάλλον. Η χρήση του Azure στη συγκεκριμένη υλοποίηση ενισχύει την αξιοπιστία του συστήματος και παρέχει επιπλέον ευελιξία για μελλοντική επέκταση ή ενσωμάτωση επιπρόσθετων υπηρεσιών. Παρακάτω απεικονίζεται το περιβάλλον εργασίας (βλπ. Εικόνα 15).



Εικόνα 15:Περιβάλλον εργασίας λογισμικού Microsoft Azure

2.3.5 Λογισμικό Arduino IDE

Το Arduino IDE είναι ένα λογισμικό ανοικτού κώδικα που αναπτύχθηκε για να προγραμματίζει πλακέτες μικροελεγκτών. Διακρίνεται για την απλότητα και την προσβασιμότητά του, προσφέροντας μια φιλική διεπαφή χρήστη με βάση μια απλοποιημένη εκδοχή της σύνταξης της C και C ++, γεγονός που το καθιστά ιδανικό για αρχάριους, εκπαιδευτικούς και επαγγελματίες στον τομέα της ηλεκτρονικής και της ανάπτυξης εφαρμογών IoT (Internet of Things). Μεταξύ των κύριων λειτουργιών του περιλαμβάνονται, ένας διαχειριστής βιβλιοθηκών για επέκταση των δυνατοτήτων των προγραμμάτων και μια σειριακή οθόνη (Serial Monitor) για επικοινωνία σε πραγματικό χρόνο με συνδεδεμένες συσκευές. Το περιβάλλον εργασίας ενσωματώνεται άρτια με το οικοσύστημα Arduino (βλπ. Εικόνα 16), προσφέροντας πρόσβαση σε μια τεράστια κοινότητα χρηστών, εκπαιδευτικά tutorials και παραδείγματα έργων, τα οποία επιταχύνουν την πρωτοτυπία και την εκμάθηση [21]. Η φιλοσοφία ανοικτού κώδικα ενθαρρύνει τη συνεργασία, επιτρέποντας στους χρήστες να τροποποιούν το λογισμικό και να συμβάλλουν στη συνεχή βελτίωσή του. Το Arduino IDE έχει απλοποιήσει την ανάπτυξη ενσωματωμένων συστημάτων, δίνοντας τη δυνατότητα σε χρήστες όλων των επιπέδων να δημιουργήσουν διαδραστικά έργα, από απλά συστήματα ελέγχου LED έως ρομποτικές εφαρμογές και συστήματα αυτοματισμού.

Εικόνα 16:Περιβάλλον εργασίας λογισμικού Arduino IDE

2.4 Σύνοψη δεύτερου κεφαλαίου

Στο Κεφάλαιο 2 παρουσιάσαμε το θεωρητικό υπόβαθρο που πλαισίωσε τη διπλωματική εργασία, με έμφαση στα βασικά τεχνολογικά και λογισμικά εργαλεία που αξιοποιήθηκαν στο σύστημα. Αρχικά, αναλύσαμε την έννοια των ενσωματωμένων συστημάτων, με επίκεντρο τους μικροελεγκτές, την αρχιτεκτονική και τη λειτουργία τους, καθώς και τις εναλλακτικές πλατφόρμες, όπως τα Single-Board Computers και τα Field-Programmable Gate Arrays (FPGAs). Στη συνέχεια, εξετάστηκαν αναλυτικά τα βασικά πρωτόκολλα επικοινωνίας του ESP32, όπως τα SPI, UART, I2S και Wi-Fi (IEEE 802.11), προκειμένου να κατανοηθεί η αλληλεπίδραση του μικροελεγκτή με τα λοιπά υποσυστήματα. Τέλος, περιγράφηκαν τα λογισμικά εργαλεία που χρησιμοποιήθηκαν σε κάθε στάδιο της ανάπτυξης, συμπεριλαμβανομένων των Autodesk Fusion 360 και EasyEDA για τον σχεδιασμό CAD και PCB, του Creality Print για την εκτύπωση 3D εξαρτημάτων, του Microsoft Azure για τις cloud υπηρεσίες και του Arduino IDE για τον προγραμματισμό και την υλοποίηση του κώδικα. Το κεφάλαιο αυτό παρείχε τις απαραίτητες θεωρητικές γνώσεις που υποστήριξαν τον σχεδιασμό και την υλοποίηση της εφαρμογής.

Κεφάλαιο 3:Προδιαγραφές-Σχεδίαση και υλοποίηση Συσκευής

Στο παρόν κεφάλαιο παρουσιάζεται αναλυτικά η διαδικασία σχεδίασης και κατασκευής της ενσωματωμένης συσκευής που αναπτύχθηκε στο πλαίσιο της διπλωματικής εργασίας. Περιγράφονται τα επιμέρους υποσυστήματα του υλικού και τα εργαλεία που χρησιμοποιήθηκαν για την ολοκλήρωση της υλοποίησης. Ιδιαίτερη έμφαση δόθηκε στην επιλογή των κατάλληλων εξαρτημάτων, όπως ο μικροελεγκτής, το υποσύστημα κάμερας και οι περιφερειακές μονάδες, καθώς και στη μεθοδολογία ολοκλήρωσης του κυκλώματος και της πλακέτας. Παράλληλα, αναλύθηκε ο σχεδιασμός του περιβλήματος της συσκευής μέσω λογισμικού CAD, καθώς και η διαδικασία τρισδιάστατης εκτύπωσης. Στόχος του κεφαλαίου ήταν να τεκμηριωθεί πλήρως η ανάπτυξη της συσκευής από τη θεωρητική σύλληψη έως το λειτουργικό πρωτότυπο.

3.1 Προδιαγραφές Συσκευής

Η επιλογή των επιμέρους ηλεκτρονικών εξαρτημάτων της συσκευής πραγματοποιήθηκε με γνώμονα τη βέλτιστη απόδοση, τη χαμηλή κατανάλωση ισχύος, τη λειτουργική πληρότητα και το χαμηλό κόστος υλοποίησης. Κάθε μονάδα που ενσωματώνεται στο σύστημα εξετάστηκε ως προς τις τεχνικές της δυνατότητες, τη συμβατότητα με το υπόλοιπο κύκλωμα και τη συνεισφορά της στη συνολική λειτουργικότητα της συσκευής. Ιδιαίτερη έμφαση δόθηκε σε παράγοντες όπως η επεκτασιμότητα, η ευκολία ενσωμάτωσης, η αξιοπιστία και η διαθεσιμότητα στην αγορά. Ακολουθεί αναλυτική παρουσίαση των βασικών κριτηρίων επιλογής και των τεχνολογικών λύσεων που υιοθετήθηκαν για κάθε επιμέρους τμήμα του συστήματος, όπως ο μικροελεγκτής, η μονάδα κάμερας, ο ενισχυτής ήχου και η οθόνη απεικόνισης.

Επιλογή Μικροελεγκτή: Κατά την αξιολόγηση των απαιτήσεων του συστήματος, κρίθηκε απαραίτητο ο μικροελεγκτής που θα επιλεγεί να πληροί ορισμένα κρίσιμα κριτήρια, όπως:

- επαρκή αριθμό ψηφιακών και αναλογικών εισόδων/εξόδων για τη διασύνδεση περιφερειακών,
- υποστήριξη πολλαπλών σειριακών πρωτοκόλλων επικοινωνίας (UART, SPI, I2S),
- δυνατότητες ασύρματης επικοινωνίας Wi-Fi,

- ικανοποιητική υπολογιστική ισχύ για την εκτέλεση σύνθετων διεργασιών,
- και χαμηλό κόστος, ώστε να είναι οικονομικά βιώσιμη η υλοποίηση.

Με βάση τα παραπάνω, εξετάστηκαν τέσσερις μικροελεγκτές:
- το Blue Pill (STM32F103C8T6) [24],
- το Arduino Nano [25],
- το Raspberry Pi 4 [26],
- και το ESP32 (Espressif Systems) [5].

Η συγκριτική αξιολόγηση των παραπάνω οδήγησε στα εξής συμπεράσματα:
- Οι **Arduino Nano** και **Blue Pill**, παρότι αποτελούν αξιόπιστες λύσεις, δεν διαθέτουν ενσωματωμένη υποστήριξη για ασύρματη επικοινωνία, γεγονός που θα απαιτούσε την προσθήκη εξωτερικών modules, αυξάνοντας την πολυπλοκότητα και το κόστος του συστήματος.

- Το **Raspberry Pi 4** προσφέρει υψηλή υπολογιστική ισχύ και πλήρεις δυνατότητες ασύρματης επικοινωνίας (Wi-Fi). Ωστόσο, το σημαντικά υψηλότερο κόστος (~35€), η αυξημένη κατανάλωση ισχύος και οι ανάγκες ψύξης το καθιστούν λιγότερο κατάλληλο για φορητές ή ενεργειακά αποδοτικές εφαρμογές.

- Το **ESP32** προσφέρει τον πλέον ιδανικό συνδυασμό χαρακτηριστικών, καθώς διαθέτει:
  - διπλό πυρήνα με συχνότητα έως 240 MHz,
  - ενσωματωμένο Wi-Fi,
  - πλούσιο αριθμό GPIOs (έως 34 pins),
  - υποστήριξη SPI, UART, PWM, ADC, DAC,
  - χαμηλό κόστος (~5,5€).

Λαμβάνοντας υπόψη όλα τα παραπάνω, επιλέχθηκε το **ESP32** ως ο καταλληλότερος μικροελεγκτής, χάρη στον άριστο συνδυασμό επιδόσεων, ευελιξίας, ενεργειακής αποδοτικότητας και ενσωματωμένης συνδεσιμότητας.

Επιλογή Μονάδας Επεξεργασίας και Μετάδοσης Εικόνας: Για τις ανάγκες λήψης, επεξεργασίας και ασύρματης μετάδοσης εικόνας, ήταν απαραίτητο να χρησιμοποιηθεί μια μονάδα που να:
- ενσωματώνει κάμερα,
- υποστηρίζει Wi-Fi,
- διαθέτει επαρκή υπολογιστική ισχύ,
- έχει μικρό μέγεθος, και
- προσφέρεται σε χαμηλό κόστος.

Η **ESP32-CAM** πληροί όλα τα παραπάνω κριτήρια, καθώς αποτελεί μία ολοκληρωμένη λύση, βασισμένη στο ισχυρό SoC ESP32 της Espressif, και προσφέρει:
- ενσωματωμένη κάμερα OV2640 ανάλυσης έως 2 Megapixels,
- δυνατότητα real-time μετάδοσης εικόνας μέσω Wi-Fi,
- υψηλή υπολογιστική ισχύ (240 MHz, dual-core), επαρκή για εφαρμογές όπως υπολογιστική όραση,
- υποδοχή για microSD κάρτα για αποθήκευση δεδομένων,
- μικρό μέγεθος, ενσωματωμένη και εξωτερική κεραία,
- πολύ χαμηλό κόστος (~6€).

Η χρήση της ESP32-CAM καταργεί την ανάγκη για ξεχωριστές μονάδες κάμερας, Wi-Fi και αποθήκευσης, μειώνοντας έτσι την πολυπλοκότητα του κυκλώματος. Η πλήρης συμβατότητά της με την κύρια μονάδα ελέγχου (ESP32) διευκολύνει την ανάπτυξη λογισμικού και την ενσωμάτωσή της στο σύστημα.

Επιλογή Ενισχυτή Ήχου: Για την αναπαραγωγή ήχου, απαιτήθηκε μία μονάδα που να παρέχει καλή ποιότητα, χαμηλή κατανάλωση, εύκολη διασύνδεση με ESP32 και χαμηλό κόστος. Αφού εξετάστηκαν εναλλακτικές λύσεις (όπως PAM8403 [27], PCM5102 [28]), επελέγη ο MAX98357A, ένας ψηφιακός ενισχυτής Class-D της Maxim Integrated, για τους εξής λόγους:

- υποστηρίζει I2S ψηφιακή είσοδο, πλήρως συμβατή με ESP32,
- έχει χαμηλή κατανάλωση ενέργειας,
- διαθέτει ισχύς εξόδου έως 3W, κατάλληλη για μικρά ηχεία (4Ω),
- παρουσιάζει μικρές διαστάσεις και ευκολία ενσωμάτωσης,
- και έχει χαμηλό κόστος και ευρεία διαθεσιμότητα.

Η επιλογή του MAX98357A εξαλείφει την ανάγκη για εξωτερικό DAC ή μετατροπείς σήματος, απλοποιώντας σημαντικά το κύκλωμα και εξασφαλίζοντας καθαρό, αξιόπιστο ήχο.

Επιλογή Οθόνης: Η ανάγκη για απεικόνιση δεδομένων και διαδραστικότητα με τον χρήστη κατέστησε απαραίτητη την ενσωμάτωση οθόνης. Τα βασικά κριτήρια επιλογής ήταν:

- η επαρκής ανάλυση και διαστάσεις,
- η υποστήριξη αφής,
- η συμβατότητα με ESP32,
- η επικοινωνία μέσω SPI,
- και το χαμηλό κόστος.

Η TFT οθόνη ILI9341 [22][32] με touch panel επελέγη, καθώς διαθέτει:

- έγχρωμη οθόνη 3.2" με ανάλυση 320x240 pixels,
- ενσωματωμένο χειριστήριο ILI9341, υποστηριζόμενο από πλήθος βιβλιοθηκών (Arduino),
- ανθεκτικό resistive touch panel, για αλληλεπίδραση χωρίς φυσικά κουμπιά,
- διασύνδεση SPI, μειώνοντας τα απαραίτητα pins και αυξάνοντας την ταχύτητα,
- τάση λειτουργίας 3.3-5V, πλήρως συμβατή με ESP32,
- κόστος μεταξύ 5–7€.

Η ILI9341 συνδυάζει λειτουργικότητα, χαμηλή κατανάλωση, καλή οπτική απόδοση και δυνατότητα ανάπτυξης γραφικού περιβάλλοντος χρήστη (GUI - Graphical user interface), καλύπτοντας πλήρως τις απαιτήσεις της εφαρμογής.

Το σύστημα αποτελεί μια ολοκληρωμένη ενσωματωμένη πλατφόρμα μικροελεγκτών και περιφερειακών εξαρτημάτων, ειδικά σχεδιασμένη για εφαρμογές υπολογιστικής όρασης, με δυνατότητες προβολής, ήχου και αλληλεπίδρασης με τον χρήστη. Οι βασικές τεχνικές προδιαγραφές περιγράφονται αναλυτικά παρακάτω (βλπ. Πίνακας 3):

1. Μικροελεγκτές

- **ESP32** [5]: Αποτελεί την κύρια μονάδα επεξεργασίας. Είναι διπλού πυρήνα, υποστηρίζει Wi-Fi, προσφέροντας υψηλές επιδόσεις και δυνατότητες επικοινωνίας σε πραγματικό χρόνο.
- **ESP32-CAM** [6]: Συμπληρωματική μονάδα με ενσωματωμένη κάμερα, κατάλληλη για μετάδοση εικόνας μέσω Wi-Fi. Ιδανική για εφαρμογές υπολογιστικής όρασης, όπως αναγνώριση εικόνας.

2. Οθόνη

- **TFT Οθόνη 3,2 ιντσών** [22][32]: Έγχρωμη οθόνη τύπου TFT LCD με ανάλυση 240x320 pixels. Διαθέτει λειτουργία αφής (touch screen), επιτρέποντας αλληλεπίδραση χρήστη μέσω γραφικών μενού, εικονικών πλήκτρων ή δυναμικών στοιχείων διεπαφής. Η σύνδεση με το ESP32 γίνεται μέσω διαύλου SPI, ενώ η αφής υλοποιείται με χρήση resistive touch controller (όπως ο XPT2046).

3. Ήχος

- **Ηχείο 3W, 4Ω**: Μικρού μεγέθους ηχείο με ικανοποιητική απόδοση ήχου.
- **Ψηφιακός Ενισχυτής MAX98357A** [23]: Μονοκαναλικός I2S ενισχυτής ήχου, πλήρως συμβατός με το ESP32. Επιτρέπει ψηφιακή αναπαραγωγή ήχου χωρίς τη χρήση επιπλέον DAC, καθιστώντας το ιδανικό για αναπαραγωγής φωνής ή ήχου.

4. Στοιχεία Συνδεσιμότητας & Ελέγχου

- **Pin Headers**: Χρησιμοποιούνται για εύκολη διασύνδεση των επιμέρους μονάδων μέσω custom PCB.
- **Momentary Buttons** (Push Buttons): Παρέχουν χειροκίνητη είσοδο και έλεγχο λειτουργιών του συστήματος (π.χ. ενεργοποίηση κάμερας, πλοήγηση σε μενού).
- **Διακόπτης ON-OFF**: Χρησιμοποιείται για τον γενικό έλεγχο τροφοδοσίας του κυκλώματος.
- **Ποτενσιόμετρο 20ΚΩ**: Επιτρέπει χειροκίνητη ρύθμιση της έντασης ήχου του ηχείου.
- **Καλώδια τύπου 30AWG**: Ιδανικά για συνδέσεις χαμηλού ρεύματος, ευέλικτα και λεπτά για εύκολη διαχείριση στο PCB.
- **Συνδέσεις τύπου JST 2.54mm**: Προσφέρουν ασφαλή και γρήγορη σύνδεση/αποσύνδεση μονάδων, με σταθερότητα στη μετάδοση σήματος και ρεύματος.
- **DC Jack 5.5x2.5mm**: Τυπική είσοδος για εξωτερικό τροφοδοτικό ή για φόρτιση μπαταριών, κατάλληλη για φορητές ενσωματωμένες εφαρμογές.
- **Πλακέτα PCB**: Προσαρμοσμένο εκτυπωμένο κύκλωμα σχεδιασμένο για να φιλοξενεί όλα τα παραπάνω εξαρτήματα με τρόπο λειτουργικό και εργονομικό.

5. Τροφοδοσία

- **Μπαταρίες NiMH 1.2V AA** (2200mAh): Επαναφορτιζόμενες, ασφαλείς και φιλικές προς το περιβάλλον, κατάλληλες για εφαρμογές μέσης κατανάλωσης.
- **Θήκη Μπαταριών 4x AA**: Υποστηρίζει σειρά σύνδεσης τεσσάρων μπαταριών, αποδίδοντας συνολικά 4.8V, επαρκή για τη λειτουργία του ESP32 και των περιφερειακών.

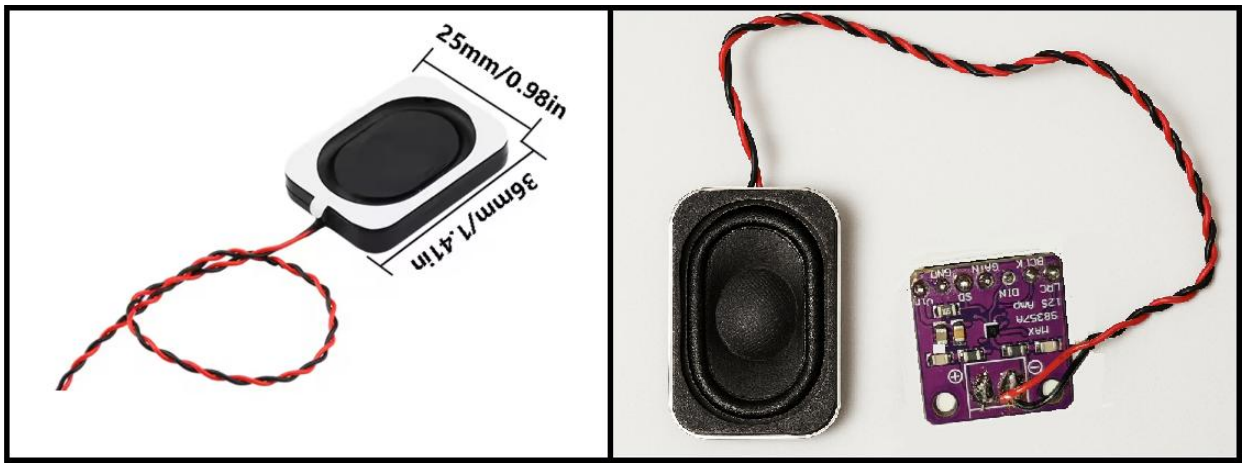
Πίνακας 3:Πίνακας βασικών τεχνικών προδιαγραφών συσκευής

Χαρακτηριστικά	Συσκευή διπλωματικής
Επεξεργαστής (CPU)	ESP32 + ESP32 CAM (Xtensa LX6 dual-core)
Μνήμη (RAM)	520 KB SRAM (+ PSRAM για ESP32 CAM)
Αποθηκευτικός Χώρος	4 MB Flush Rom SPI
Οθόνη	3.2" TFT οθόνη αφής
Κάμερα	OV2640 (2MP)
Συνδεσιμότητα	Wi-Fi
Αισθητήρες	Δεν καθορίζεται
Ήχος	Μονοφωνικό ηχείο 3W 4Ω
Διαστάσεις (ΜxΠxΥ)	115mm x 67mm x 57mm
Βάρος	225g
Μπαταρία	4x 2200mAh Nimh
Λειτουρ. Σύστημα	Δεν καθορίζεται
Διασύνδ. Χρήστη	Οθόνη αφής
Τιμή	$45 - (5,000 free image analises/ μήνα)

3.2 Ηλεκτρονικά εξαρτήματα

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, σχεδιάστηκε και κατασκευάστηκε μία ενσωματωμένη συσκευή βασισμένη στον μικροελεγκτή ESP32, ενσωματώνοντας πληθώρα ηλεκτρονικών εξαρτημάτων για την υλοποίηση προηγμένων λειτουργιών. Συγκεκριμένα, χρησιμοποιήθηκαν δύο εκδόσεις του ESP32, το ESP32-WROOM-32 και το ESP32-CAM. Αμφότερες οι εκδόσεις διαθέτουν ενσωματωμένη συνδεσιμότητα Wi-Fi, με το ESP32-CAM να προσφέρει επιπλέον τη δυνατότητα λήψης εικόνας και βίντεο μέσω του ενσωματωμένου αισθητήρα OV2640. Για την ηχητική έξοδο επιλέχθηκε ο ψηφιακός ενισχυτής ήχου MAX98357, ο οποίος μετατρέπει σήματα I2S σε αναλογικό ήχο, επιτρέποντας την απευθείας σύνδεση ηχείου. Περιλαμβάνεται επίσης ποτενσιόμετρο, το οποίο χρησιμοποιείται για τη ρύθμιση μεταβλητών παραμέτρων λειτουργίας της συσκευής, όπως ένταση ήχου, ενώ ο μετρητής τάσης (voltage checker) εξασφαλίζει την παρακολούθηση της τροφοδοσίας και την αποφυγή υποτάσεων. Η διεπαφή με τον χρήστη επιτυγχάνεται μέσω έγχρωμης οθόνης αφής TFT 3.2 ιντσών, η οποία επιτρέπει όχι μόνο την απεικόνιση δεδομένων αλλά και τον άμεσο χειρισμό της συσκευής. Όλα τα επιμέρους υποσυστήματα συνδέονται μέσω συνδέσμων τύπου JST 2.56 mm, οι οποίοι προσφέρουν αξιόπιστες και ασφαλείς συνδέσεις μεταξύ πλακετών και εξαρτημάτων. Η ολοκλήρωση αυτών των στοιχείων σε ένα λειτουργικό και ολοκληρωμένο σύστημα απαίτησε λεπτομερή σχεδίαση του κυκλώματος και της πλακέτας (PCB). Στη συνέχεια, παρουσιάζονται αναλυτικά τα τεχνικά χαρακτηριστικά των επιμέρους ηλεκτρονικών εξαρτημάτων που χρησιμοποιήθηκαν.

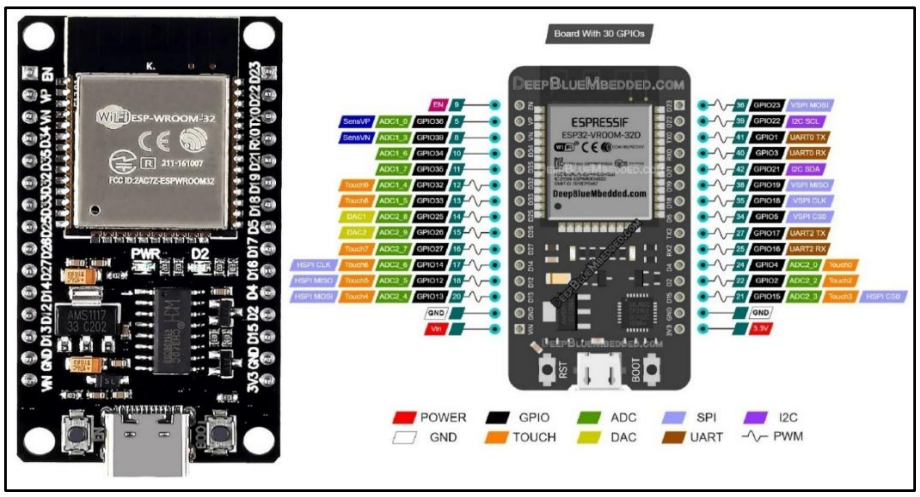
1. **Ηχείο 4Ω 25x36 3W:** Το ηχείο είναι ένα μικρών διαστάσεων αλλά ισχυρό ηχητικό εξάρτημα, ιδανικό για ενσωμάτωση σε φορητές και συμπαγείς ηλεκτρονικές διατάξεις. Παρά το μικρό του μέγεθος (25x36 χιλιοστά), προσφέρει ισχύ 3 Watt και αντίσταση 4Ω, επαρκή για καθαρή και δυνατή αναπαραγωγή ήχου. Η σύνδεση του πραγματοποιείται μέσω διπλού καλωδίου με απ' ευθείας συγκόλληση με τον ενισχυτή ήχου MAX98357A (βλπ. Εικόνα 17).



Εικόνα 17:Ηχείο 4Ohm 3W

2. **ESP32:** Το ESP32-WROOM-32 [5][29] αποτελεί ένα από τα πλέον διαδεδομένα και ευέλικτα modules της εταιρείας Espressif Systems, κατάλληλο για ένα ευρύ φάσμα εφαρμογών που περιλαμβάνουν το Διαδίκτυο των Πραγμάτων (IoT), ασύρματη επικοινωνία, αυτοματισμούς και ενσωματωμένα συστήματα (βλπ. Εικόνα 18). Το module βασίζεται στον διπύρηνο επεξεργαστή Tensilica Xtensa LX6, ο οποίος λειτουργεί σε συχνότητες έως και 240 MHz, προσφέροντας υψηλή υπολογιστική ισχύ και δυνατότητες παράλληλης εκτέλεσης διεργασιών. Η τάση λειτουργίας των ακίδων εισόδου/εξόδου (I/O) είναι 3.3V, ενώ το εύρος θερμοκρασίας λειτουργίας κυμαίνεται από -40°C έως +85°C, καθιστώντας το κατάλληλο για εφαρμογές σε απαιτητικά περιβάλλοντα. Σε επίπεδο μνήμης, το ESP32

διαθέτει 448 KB ROM, η οποία προορίζεται για βασικές ρουτίνες συστήματος και εκκίνησης, και 520 KB SRAM, κατανεμημένα για χρήση από τον επεξεργαστή και τα περιφερειακά. Επιπλέον, το module περιλαμβάνει 4 MB εξωτερικής SPI Flash, απαραίτητα για την αποθήκευση του firmware και λοιπών σταθερών δεδομένων. Σημειώνεται ότι η παρούσα έκδοση δεν υποστηρίζει PSRAM, για εφαρμογές που απαιτούν αυξημένη δυναμική μνήμη προτείνεται η χρήση του ESP32-WROVER, το οποίο διαθέτει ενσωματωμένη εξωτερική PSRAM.



Εικόνα 18:ESP32 Pinout
Πηγή:https://deepbluembedded.com/getting-started-with-esp32-programming-tutorials/

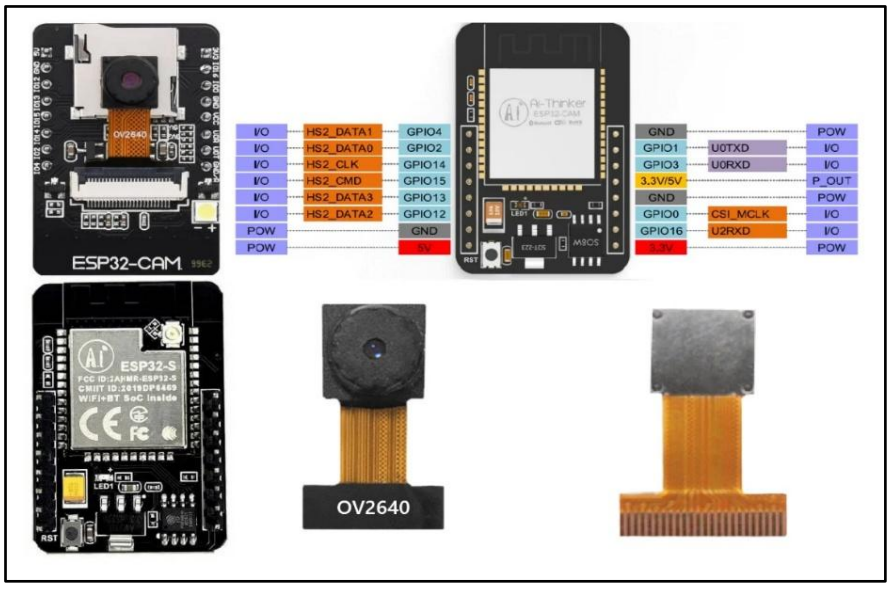
Το module ενσωματώνει ασύρματες δυνατότητες Wi-Fi 802.11 b/g/n στη ζώνη των 2.4 GHz, με υποστήριξη λειτουργιών σταθμού (STA), σημείου πρόσβασης (AP) ή συνδυαστικά (STA+AP), καθώς και Wi-Fi Direct για απευθείας επικοινωνία μεταξύ συσκευών. Επιπρόσθετα, υποστηρίζονται σύγχρονα πρωτόκολλα ασφάλειας (WPA, WPA2, WPA3), διασφαλίζοντας την εμπιστευτικότητα και την ακεραιότητα της μεταδιδόμενης πληροφορίας. Πέραν του Wi-Fi, υποστηρίζεται και Bluetooth v4.2, τόσο στην κλασική του μορφή (BR/EDR) όσο και στο πρότυπο Bluetooth Low Energy (BLE), με λειτουργίες GATT server/client, BLE broadcasting και Serial Port Profile (SPP) για ασύρματη σειριακή επικοινωνία με έξυπνες συσκευές. Το ESP32 διαθέτει έως 34 προγραμματιζόμενα GPIOs γενικής χρήσης, τα οποία υποστηρίζουν πλειάδα περιφερειακών λειτουργιών, όπως 16 κανάλια ADC 12-bit, 2 DAC 8-bit, 3 θύρες UART, 4 διαύλους SPI, 2 I2C, καθώς και 2 διαύλους I2S για εφαρμογές streaming ήχου. Όλα τα GPIOs υποστηρίζουν PWM, καθιστώντας δυνατή τη διαχείριση κινητήρων ή φωτεινότητας LED. Επίσης, υποστηρίζεται σύνδεση με κάρτα SD μέσω διεπαφής SDIO, ενώ διαθέτει 10 capacitive pins για είσοδο αφής, αισθητήρα Hall, και αισθητήρα θερμοκρασίας, προσφέροντας ενισχυμένες δυνατότητες παρακολούθησης και ελέγχου.

Στον τομέα της ασφάλειας, ο μικροελεγκτής υποστηρίζει λειτουργίες Secure Boot και κρυπτογράφηση flash, ενώ ενσωματώνει επιτάχυνση υλικού (hardware acceleration) για αλγορίθμους AES, SHA, RSA, ECC, καθώς και γεννήτρια τυχαίων αριθμών (RNG). Αναφορικά με την ενεργειακή απόδοση, παρουσιάζει προφίλ κατανάλωσης: περίπου 160–240 mA σε ενεργό ρόλο (Wi-Fi και CPU), 3–20 mA σε κατάσταση modem-sleep, 0.8 mA σε light-sleep, 10 μA σε deep-sleep, και λιγότερο από 5 μA σε hibernation, προσφέροντας εξαιρετικές δυνατότητες για φορητές ή ενεργειακά αποδοτικές εφαρμογές. Τέλος, το module έχει διαστάσεις 52 mm x 28mm και παρέχεται σε μορφή SMD (surface-mount device) με 30 ακίδες (pins), επιτρέποντας την εύκολη ενσωμάτωσή του σε σχεδιάσεις πλακετών (PCB) περιορισμένου μεγέθους, διατηρώντας παράλληλα υψηλό βαθμό λειτουργικότητας και διασυνδεσιμότητας. Ακολουθεί συνοπτικός Πίνακας με τα τεχνικά χαρακτηριστικά (βλπ. Πίνακας 4).

Πίνακας 4:Συνοπτικός πίνακας τεχνικών χαρακτηριστικών ESP32 WROOM32

ESP32-WROOM-32	
Κατηγορία	**Περιγραφή**
Μικροελεγκτής	ESP32-D0WDQ6
Πυρήνες CPU	2 x Xtensa LX6 32-bit
Συχνότητα λειτουργίας	Έως 240 MHz
Εσωτερική ROM	448 KB
SRAM	520 KB + RTC FAST/SLOW RAM (2 x 8 KB)
Εξωτερική Flash	4 MB SPI (τυπικά)
PSRAM	Δεν υποστηρίζεται
Wi-Fi	802.11 b/g/n (2.4 GHz), STA / AP / STA+AP, WPA/WPA2/WPA3
Bluetooth	v4.2 BR/EDR & BLE (GATT server/client, SPP)
GPIOs	Έως 34 (πολλά επαναχαρτογραφήσιμα)
ADC	Έως 18 κανάλια 12-bit (ADC1 & ADC2)
DAC	2 x 8-bit
UART	3 (2 πλήρως λειτουργικές)
SPI	4 (HSPI, VSPI, κ.λπ.)
I2C	2
I2S	2
PWM	Υποστήριξη σε όλα τα GPIOs
Touch sensors	10 χωρητικοί αισθητήρες
Hall Sensor	Ναι
Αισθητήρας θερμοκρασίας	Ναι (εσωτερικός)
Ασφάλεια	Secure Boot, Flash Encryption, AES, SHA, RSA, ECC, RNG (hardware acceleration)
Τάση λειτουργίας I/O	3.3V
Κατανάλωση (τυπική)	160–240 mA (Active), 3–20 mA (Modem Sleep), ~0.8 mA (Light Sleep), <10 μA (Deep Sleep)
Διαστάσεις	18.0 mm x 25.5 mm x 3.1 mm
Μορφή	SMD (Surface Mount Device), 38 ακίδες
Θερμοκρασία λειτουργίας	-40°C έως +85°C

3. **ESP32 CAM**: Το ESP32-CAM [6][7] αποτελεί μια ολοκληρωμένη λύση ενσωματωμένου συστήματος που συνδυάζει τον ισχυρό μικροελεγκτή ESP32-D0WD, σχεδιασμένο από την AI-Thinker με δυνατότητες ασύρματης επικοινωνίας και ενσωματωμένη υποστήριξη κάμερας. Το module είναι ιδανικό για εφαρμογές όπως συστήματα επιτήρησης, αναγνώριση προσώπου, έξυπνες πόρτες, έλεγχος πρόσβασης και άλλες λύσεις που βασίζονται σε οπτικά δεδομένα (βλπ. Εικόνα 19).



Εικόνα 19:ESP32 CAM Pinout
Πηγή:https://diyprojectslabs.com/guide-to-using-the-esp32-cam-module/

Στην καρδιά της μονάδας βρίσκεται ο διπύρηνος μικροελεγκτής ESP32-D0WD, ο οποίος βασίζεται στην αρχιτεκτονική Tensilica Xtensa LX6 32-bit και λειτουργεί σε συχνότητες έως και 240 MHz. Διαθέτει 520 KB SRAM, 4 MB εξωτερική SPI Flash για την αποθήκευση του firmware, καθώς και 4 MB ενσωματωμένη PSRAM, προσφέροντας την απαραίτητη μνήμη για απαιτητικές εργασίες επεξεργασίας εικόνας και βίντεο.

Το module συνοδεύεται από τον αισθητήρα κάμερας OV2640 [8], έναν CMOS αισθητήρα 2 Megapixel, ο οποίος υποστηρίζει αναλύσεις από 160x120 (QQVGA) έως και 1600x1200 (UXGA). Ο αισθητήρας διαθέτει δυνατότητα JPEG συμπίεσης, γεγονός που επιτρέπει την αποδοτική μετάδοση εικόνων μέσω Wi-Fi. Η κάμερα συνδέεται στο ESP32-CAM μέσω 24-pin FPC connector και ελέγχεται μέσω DVP interface. Όσον αφορά την ασύρματη συνδεσιμότητα, το ESP32-CAM υποστηρίζει Wi-Fi 802.11 b/g/n (2.4 GHz) και Bluetooth v4.2, τόσο σε μορφή BR/EDR όσο και BLE, επιτρέποντας τη δημιουργία δικτυακών εφαρμογών όπως web-based IP camera ή έξυπνο σύστημα καταγραφής, απομακρυσμένο έλεγχο, streaming και αποθήκευσης εικόνας. Επιπλέον, διαθέτει υποδοχή για κάρτα microSD, μέσω της οποίας είναι δυνατή η τοπική αποθήκευση αρχείων εικόνας ή βίντεο.

Το module περιλαμβάνει επίσης ένα ισχυρό λευκό LED που λειτουργεί ως φλας και είναι συνδεδεμένο στο GPIO4, γεγονός που το καθιστά ιδιαίτερα χρήσιμο για λήψεις σε συνθήκες χαμηλού φωτισμού. Αν και σημαντικός αριθμός ακίδων GPIO δεσμεύεται εσωτερικά από την κάμερα OV2640 και τη μνήμη PSRAM, η διαμόρφωση των ακίδων (pinout) προσφέρει επαρκές πλήθος διαθέσιμων GPIOs για βασικές λειτουργίες εισόδου/εξόδου. Συγκεκριμένα, υποστηρίζονται διασυνδέσεις όπως UART, PWM, I2C και SPI, με δυνατότητα επαναχαρτογράφησης (remapping) των λειτουργιών στις ελεύθερες ακίδες. Τα GPIO0, GPIO1, GPIO3, GPIO4, GPIO12 έως GPIO16, καθώς και τα GPIO33 έως GPIO35, είναι προσβάσιμα για γενική χρήση, ενώ τα GPIO6 έως GPIO11 είναι δεσμευμένα για τη λειτουργία της ενσωματωμένης SPI Flash και δεν είναι διαθέσιμα για εξωτερικές συνδέσεις.

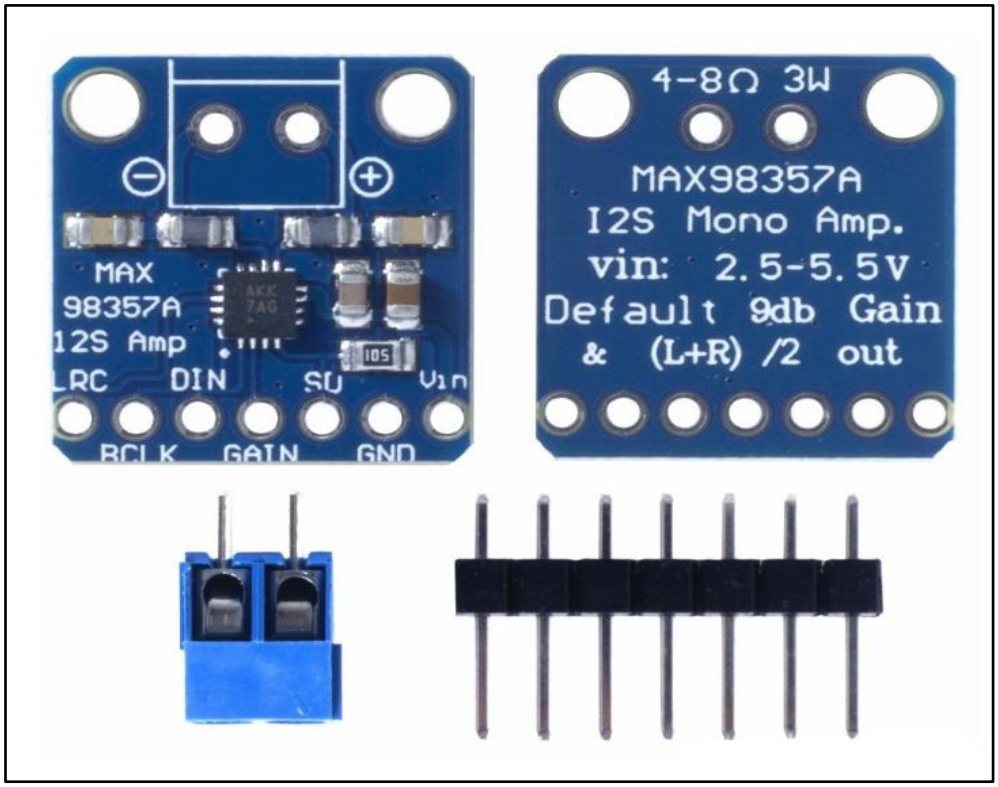
Το ESP32-CAM δεν διαθέτει ενσωματωμένο USB-to-Serial μετατροπέα, και για την προγραμματισμό του απαιτείται εξωτερικός μετατροπέας (όπως FTDI ή ESP32 CAM Motherboard). Η τροφοδοσία του ESP32-CAM γίνεται με 5V (μέσω VIN pin ή USB), ενώ το κύκλωμα διαθέτει ενσωματωμένο ρυθμιστή τάσης 3.3V. Σε ενεργή λειτουργία Wi-Fi και κάμερας, η κατανάλωση μπορεί να φτάσει έως 240 mA, γεγονός που απαιτεί σταθερή παροχή ρεύματος.

Τέλος, το module έχει πολύ μικρές διαστάσεις (περίπου 27 x 40.5 mm), καθιστώντας το ιδανικό για φορητές και ενσωματωμένες εφαρμογές, ενώ υποστηρίζει δυνατότητες όπως Over-the-Air (OTA) ενημερώσεις, λειτουργία Web Server, και αναγνώριση προσώπου μέσω του framework ESP-WHO της Espressif. Ακολουθεί συνοπτικός Πίνακας με τα τεχνικά χαρακτηριστικά (βλπ. Πίνακα 5).

Πίνακας 5:Συνοπτικός πίνακας τεχνικών χαρακτηριστικών ESP32 CAM

ESP32-CAM	
Κατηγορία	**Περιγραφή**
Μικροελεγκτής	ESP32-D0WD
Συχνότητα CPU	Έως 240 MHz
SRAM	520 KB
Flash	4 MB SPI Flash
PSRAM	4 MB
Wi-Fi	802.11 b/g/n
Bluetooth	v4.2 BR/EDR + BLE
Αισθητήρας κάμερας	OV2640, 2MP, JPEG/YUV, έως UXGA
Υποδοχή microSD	Ναι, με υποστήριξη SDIO
Διαθέσιμα GPIOs	GPIO 0, 1, 3, 4, 12–16, 33–35
LED Flash	Ναι (GPIO4)
UART	1
I2C/SPI	Υποστήριξη μέσω remapping
Τροφοδοσία	5V μέσω VIN ή USB programmer
Κατανάλωση	Έως 240 mA
Διαστάσεις	27 mm x 40.5 mm
USB-to-Serial	Όχι – απαιτεί εξωτερικό FTDI, CP2102 ή ESP32-CAM-MB

4. **MAX98357A**: Ο Adafruit MAX98357A [23][30] είναι ένας ψηφιακός ενισχυτής μονού καναλιού (mono) κλάσης D, σχεδιασμένος για την άμεση μετατροπή ψηφιακού σήματος τύπου I2S σε ενισχυμένο αναλογικό σήμα, κατάλληλο για την οδήγηση ηχείων (βλπ. Εικόνα 20). Βασίζεται στο ολοκληρωμένο κύκλωμα MAX98357A, το οποίο υποστηρίζει ανάλυση 24-bit και συχνότητες δειγματοληψίας από 8 kHz έως 96 kHz, παρέχοντας υψηλής ποιότητας αναπαραγωγή ήχου. Η ισχύς εξόδου φτάνει έως και 3.2W RMS σε φορτίο 4Ω με τροφοδοσία 5V, ενώ το ποσοστό παραμόρφωσης (THD+N) διατηρείται κάτω από 1%, προσφέροντας καθαρό και αποδοτικό ήχο.
   Η είσοδος ήχου πραγματοποιείται μέσω των ακροδεκτών DIN (Data In), BCLK (Bit Clock) και LRCLK (Left/Right Clock), οι οποίοι λειτουργούν σε λογικά επίπεδα 3.3V ή 5V, ακολουθώντας το πρότυπο I2S. Ο ενισχυτής ενσωματώνει αυτόματη ανίχνευση του φορμά δεδομένων και συγχρονισμού, καθιστώντας τον εξαιρετικά εύκολο στη χρήση, χωρίς την ανάγκη για εξωτερική παραμετροποίηση ή λογισμική ρύθμιση. Επιπλέον, δεν απαιτεί σήμα MCLK, γεγονός που τον καθιστά πλήρως συμβατό με μικροελεγκτές όπως το ESP32 και το Raspberry Pi, τα οποία συχνά δεν παρέχουν τέτοια έξοδο.



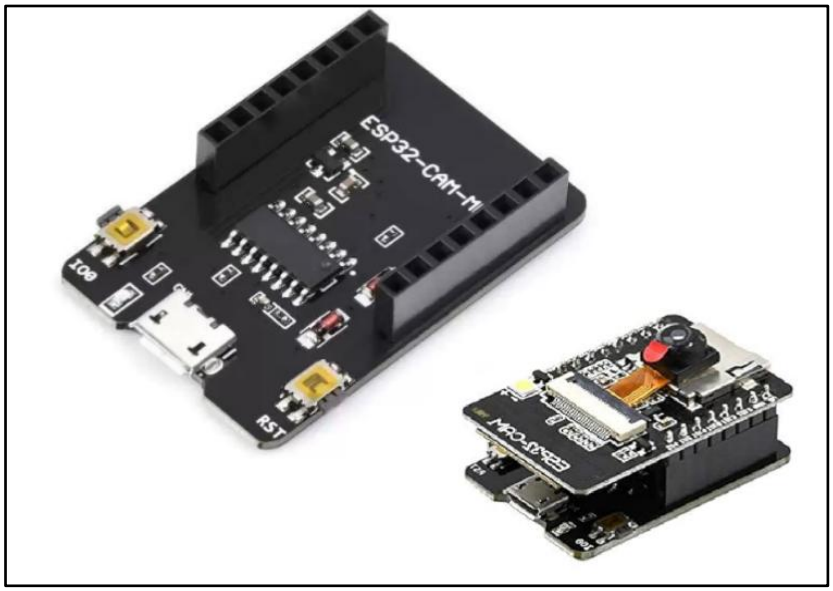
Εικόνα 20:MAX98357A - Ψηφιακός ενισχυτής μονού καναλιού (mono) κλάσης D
Πηγή: AliExpress-High Quality MAX98357 MAX98357A I2S 3W Class D Amplifier

Η πλακέτα τροφοδοτείται με τάση από 2.5V έως 5.5V μέσω των ακροδεκτών Vin και GND, με συνιστώμενη χρήση τροφοδοτικού ικανό να παρέχει τουλάχιστον 800 mA για σταθερή λειτουργία χωρίς θόρυβο ή αστάθεια. Ο ακροδέκτης GAIN επιτρέπει την επιλογή της ενίσχυσης (gain) μεταξύ 3dB, 6dB, 9dB, 12dB ή 15dB, ανάλογα με τον τρόπο σύνδεσης (π.χ. pull-up/down με αντιστάσεις). Επιπλέον, ο ακροδέκτης SD/Mode χρησιμοποιείται για τη λειτουργία ενεργοποίησης/απενεργοποίησης (shutdown), αλλά και για την επιλογή του ηχητικού καναλιού που θα αναπαραχθεί – αριστερό (Left), δεξί (Right) ή μίξη L+R/2 (μονοφωνικό). Χάρη στη μικρή του διάσταση, την ευκολία ενσωμάτωσης και τη χαμηλή κατανάλωση ενέργειας, ο MAX98357A είναι ιδανικός για χρήση σε φορητές εφαρμογές, έξυπνα ηχεία, προσωπικά projects με ESP32, και συστήματα όπου απαιτείται υψηλής ποιότητας ήχος με απλό κύκλωμα και ελάχιστο λογισμικό. Ακολουθεί συνοπτικός Πίνακας με τα τεχνικά χαρακτηριστικά (βλπ. Πίνακα 6).

Πίνακας 6:Συνοπτικός πίνακας τεχνικών χαρακτηριστικών MAX98357A

MAX98357A	
Κατηγορία	**Περιγραφή**
Τύπος	Ψηφιακός Ενισχυτής Ήχου Κλάσης D, Mono
Τσιπ	MAX98357A
Είσοδος	I2S (DIN, BCLK, LRCLK), χωρίς MCLK
Ανάλυση / Συχνότητα	Έως 24-bit, 8 kHz έως 96 kHz
Ισχύς εξόδου	3.2W RMS @ 5V / 4Ω
Τάση λειτουργίας	2.5V έως 5.5V
Επιλογή gain	3dB, 6dB, 9dB, 12dB, 15dB
Λειτουργία shutdown	Ναι (μέσω SD/Mode)
Επιλογή καναλιού	Left, Right ή (L+R)/2
Συμβατότητα	ESP32, Raspberry Pi, κ.ά.
Λογικά επίπεδα	3.3V ή 5V
Ενσωμάτωση	Απλή χρήση, χωρίς παραμετροποίησης
Διαστάσεις	Μικρές – Breakout board

5. **ESP32 CAM Motherboard**: Η πλακέτα ESP32-CAM-MB [31] αποτελεί μια εξειδικευμένη βάση προγραμματισμού για το module ESP32-CAM, ενσωματώνοντας μετατροπέα USB σε σειριακή επικοινωνία (USB-to-Serial), γεγονός που απλοποιεί σημαντικά τη διαδικασία φόρτωσης κώδικα και αποσφαλμάτωσης (βλπ. Εικόνα 21). Χάρη στον ενσωματωμένο μετατροπέα CH340G ή CP2102, επιτρέπει την απευθείας σύνδεση με υπολογιστή μέσω θύρας microUSB, προσφέροντας ταυτόχρονα και τροφοδοσία 5V στο κύκλωμα. Η πλακέτα διαθέτει ειδική υποδοχή τύπου socket για την ασφαλή τοποθέτηση του ESP32-CAM, εξασφαλίζοντας σταθερή μηχανική σύνδεση, ενώ μέσω του ενσωματωμένου κυκλώματος υποστηρίζεται αυτόματη είσοδος σε boot mode, χωρίς την ανάγκη χειροκίνητης παρέμβασης (π.χ. σύνδεσης του GPIO0 με το GND).



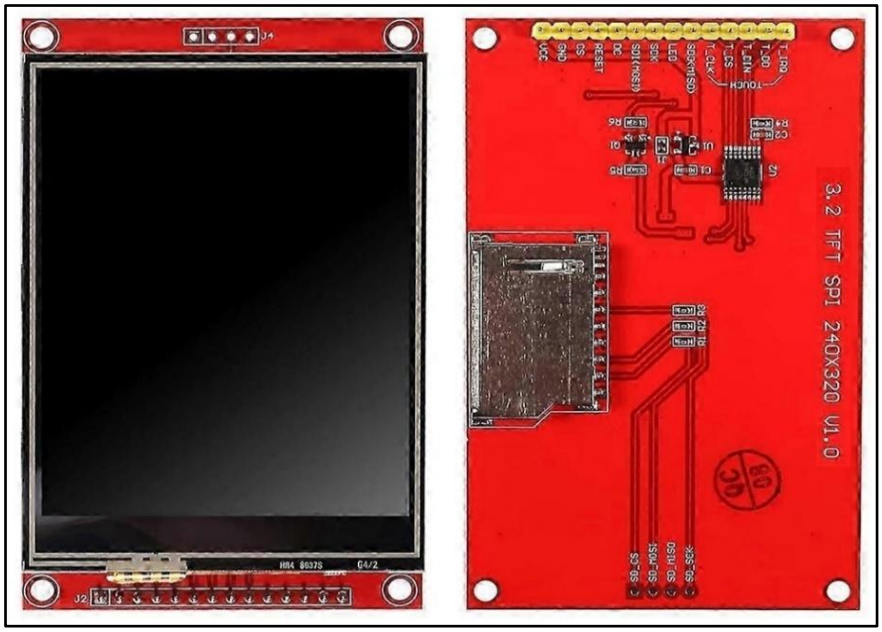
Εικόνα 21:ESP32 CAM Motherboard - Βάση προγραμματισμού με USB
Πηγή: AliExpress - ESP32-CAM ESP32-CAM-MB

Επιπλέον, ενσωματώνει κουμπί RESET, επιτρέποντας την επανεκκίνηση του μικροελεγκτή κατά τη διάρκεια της ανάπτυξης ή των δοκιμών. Ο συνολικός σχεδιασμός της ESP32-CAM-MB είναι προσανατολισμένος στην ευχρηστία και την αξιοπιστία, καθιστώντας την ιδανική επιλογή τόσο για αρχάριους όσο και για προχωρημένους χρήστες που επιθυμούν σταθερή τροφοδοσία και αξιόπιστο προγραμματισμό του ESP32-CAM. Ακολουθεί συνοπτικός Πίνακας με τα τεχνικά χαρακτηριστικά (βλπ. Πίνακα 7).

Πίνακας 7:Συνοπτικός πίνακας τεχνικών χαρακτηριστικών ESP32 CAM MB

ESP32 – CAM - MB Micro USB Programmer	
Κατηγορία	**Περιγραφή**
Λειτουργία	Βάση (motherboard) για προγραμματισμό & τροφοδοσία του ESP32-CAM
Συμβατότητα	ESP32-CAM (AI-Thinker module)
Μετατροπέας USB-UART	CH340G ή CP2102 (ανά έκδοση)
Θύρα USB	microUSB (USB 2.0, για δεδομένα + ρεύμα)
Τροφοδοσία εξόδου	5V DC μέσω USB – με εσωτερικό ρυθμιστή 3.3V
Αυτόματη είσοδος σε Boot	Ναι (δεν απαιτεί χειροκίνητη σύνδεση IO0 στο GND)
Κουμπί RESET	Ναι
Κουμπί BOOT/FLASH	Όχι, η λειτουργία καλύπτεται αυτόματα από το κύκλωμα
Υποδοχή ESP32-CAM	2x9 υποδοχή τύπου header (συμβατό pinout)
Εύκολη χρήση με Arduino	Ναι, Plug & Play με Arduino IDE ή ESP-IDF
LED ενδείξεων	Όχι (στις περισσότερες εκδόσεις)
Διαστάσεις	~35 mm x 27 mm (χωρίς το ESP32-CAM)
Χρήση	Προγραμματισμός, τροφοδοσία, debugging του ESP32-CAM

6. **TFT TOUCH ILI9341**: Η ILI9341 TFT είναι μια έγχρωμη οθόνη 3.2 ιντσών, με ανάλυση 240x320 pixels, η οποία βασίζεται στον ελεγκτή οθόνης ILI9341 και ενσωματώνει αντιστάσεως τύπου (resistive) touch panel (βλπ. Εικόνα 22). Υποστηρίζει 16-bit χρωματική παλέτα (RGB565), αποδίδοντας έως και 65.536 διαφορετικά χρώματα, προσφέροντας ευκρινή και ζωντανή απεικόνιση γραφικών, εικόνων και κειμένου. Η διασύνδεση της οθόνης με μικροελεγκτές πραγματοποιείται κυρίως μέσω SPI (Serial Peripheral Interface), γεγονός που την καθιστά συμβατή με ένα ευρύ φάσμα πλατφορμών με περιορισμένα GPIOs. Ο ελεγκτής αφής είναι συνήθως ο XPT2046, ο οποίος επικοινωνεί επίσης μέσω SPI και επιτρέπει την ανίχνευση της θέσης αφής με 12-bit ανάλυση [22][32]. Η λειτουργία αφής είναι αντιστάσεως (resistive), πράγμα που σημαίνει ότι ανταποκρίνεται σε πίεση και μπορεί να χρησιμοποιηθεί ακόμα και με γάντια ή στυλό.



Εικόνα 22:Οθόνη TFT 3.2 με ILI9341 driver
Πηγή :https://www.fruugo.gr/32inch-ili9341-spi-tft-lcd

Η οθόνη απαιτεί τροφοδοσία 3.3V για τα σήματα λογικής, αν και ορισμένες εκδόσεις της διαθέτουν εσωτερικό ρυθμιστή που επιτρέπει τροφοδοσία με 5V. Σε εφαρμογές με ESP32, χρησιμοποιείται συχνά η βιβλιοθήκη TFT_eSPI για τον ελεγκτή ILI9341 και η XPT2046_Touchscreen για την αφή, υποστηρίζοντας ταχύτατη σχεδίαση γραφικών και multi-layer GUI. Ακολουθεί συνοπτικός Πίνακας με τα τεχνικά χαρακτηριστικά (βλπ. Πίνακα 8).

Πίνακας 8:Συνοπτικός πίνακας τεχνικών χαρακτηριστικών TFT 3.2 ILI9341 driver

TFT 3.2" με ILI9341 Driver	
Χαρακτηριστικό	**Τιμή / Περιγραφή**
Μέγεθος Οθόνης	3.2 ίντσες
Ανάλυση	240 x 320 pixels (QVGA)
Driver Controller	ILI9341
Τεχνολογία Οθόνης	TFT LCD (Thin Film Transistor)
Τύπος Αφής	Resistive, ελεγκτής XPT2046
Ανάλυση αφής	12-bit ανά άξονα (4096 x 4096)
Επικοινωνία	SPI
Τάση Λειτουργίας	3.3V λογική / 5V τροφοδοσία (ανάλογα το breakout board)
Έγχρωμη Απόδοση	65K χρώματα (16-bit RGB565)
Φωτισμός (Backlight)	LED Backlight, συνήθως σταθερός ή ρυθμιζόμενος μέσω PWM
Κατανάλωση Ρεύματος	40–80 mA (ανάλογα τον φωτισμό και λειτουργία αφής)
Υποστηριζόμενες Βιβλιοθήκες	Adafruit_ILI9341, TFT_eSPI, UTFT, LovyanGFX, XPT2046_Touchscreen
Διαστάσεις Πλακέτας	Περίπου 77mm x 55mm
Συμβατότητα	ESP32, Arduino, STM32, Raspberry Pi και άλλες πλατφόρμες

7. **Εξαρτήματα συνδεσιμότητας και Ελέγχου**: Στο πλαίσιο της διπλωματικής εργασίας, χρησιμοποιήθηκαν διάφορα παθητικά και μηχανικά εξαρτήματα που συμβάλλουν στην ηλεκτρική συνδεσιμότητα και τον χειροκίνητο έλεγχο του κυκλώματος. Ανάμεσα σε αυτά περιλαμβάνονται pin headers, JST συνδέσεις, διακόπτης ON/OFF, momentary buttons και καλώδιο τύπου 30 AWG (βλπ. Εικόνα 23). Οι pin headers είναι γραμμικοί ακροδέκτες με τυπικό βήμα 2,54 mm (0.1"), που διευκολύνουν την ηλεκτρική σύνδεση μεταξύ διαφορετικών πλακετών ή μεταξύ πλακέτας και εξωτερικών στοιχείων. Παρέχονται σε αρσενική ή θηλυκή μορφή και διατίθενται για τοποθέτηση κάθετα ή οριζόντια, καθιστώντας τα ιδιαίτερα χρήσιμα για προσωρινές συνδέσεις, ανάπτυξη πρωτοτύπων και συντήρηση.

   Οι **JST XH συνδέσεις με βήμα 2,54 mm** και αριθμό επαφών από 2 έως 5 χρησιμοποιούνται για σταθερές και ασφαλείς συνδέσεις καλωδίων με πλακέτες [33]. Η στιβαρή μηχανική κατασκευή τους και η ικανότητα μεταφοράς ρεύματος έως 3A τις καθιστούν ιδανικές για εφαρμογές παροχής ισχύος ή μεταφοράς σημάτων.

   Ο **διακόπτης ON/OFF τύπου SPDT** (Single Pole, Double Throw) είναι υπεύθυνος για την ενεργοποίηση και απενεργοποίηση του κυκλώματος [34]. Αποτελείται από δύο ακροδέκτες και διαθέτει δύο σταθερές καταστάσεις: ανοικτό και κλειστό κύκλωμα, είναι τύπου toggle και τοποθετείται είτε επάνω στην πλακέτα είτε σε εξωτερικό περίβλημα. Διαχειρίζεται τυπικά τάσεις από 12V έως 250V και ρεύματα από 0,5A έως 5A, ανάλογα με την κατασκευή του.

   Τα **momentary buttons**, γνωστά και ως στιγμιαίοι διακόπτες, είναι κουμπιά που λειτουργούν μόνο όσο παραμένουν πατημένα [35]. Απελευθερώνοντάς τα, επιστρέφουν αυτόματα στην αρχική τους (ανοικτή) κατάσταση. Αυτού του τύπου οι διακόπτες χρησιμοποιούνται συνήθως για την αποστολή παλμών ελέγχου (π.χ. reset, εντολές χρήστη) και προσφέρουν απλή αλλά αξιόπιστη μηχανική διεπαφή.

   Το **ποτενσιόμετρο 20K** είναι ένας μεταβλητός αντιστάτης με ονομαστική αντίσταση 20 ΚΩ, που χρησιμοποιείται για τη ρύθμιση της έντασης σήματος ή τάσης σε ηλεκτρονικά κυκλώματα [36]. Αποτελείται από μια αντιστατική διαδρομή και έναν κινητό δρομέα που μετακινείται χειροκίνητα, μεταβάλλοντας έτσι την τιμή της αντίστασης μεταξύ των ακροδεκτών. Χρησιμοποιείται ευρέως σε εφαρμογές όπως ενισχυτές ήχου, φωτιστικά dimmers, και ρυθμιστές ταχύτητας κινητήρων. Η επιλογή ενός ποτενσιόμετρου 20K

εξαρτάται από τις απαιτήσεις του κυκλώματος, και προσφέρει ακρίβεια στον έλεγχο του ρεύματος ή της τάσης.

Εικόνα 23:Εξαρτήματα συνδεσιμότητας και ελέγχου

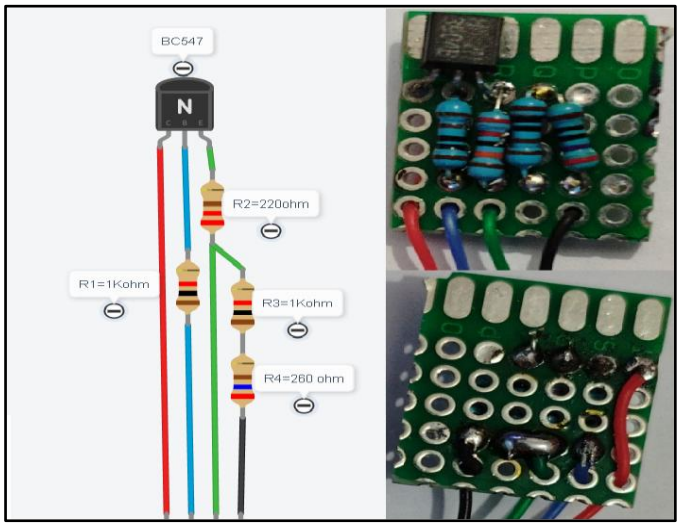
Το **DC jack 5.5x2.5** είναι ένας κοινός τύπος υποδοχής συνεχούς ρεύματος (DC) που χρησιμοποιείται για την τροφοδοσία ηλεκτρονικών συσκευών αλλά και για την φόρτιση μπαταριών [37]. Οι διαστάσεις του αναφέρονται στη διάμετρο του εξωτερικού (5.5 χιλιοστά) και του εσωτερικού (2.5 χιλιοστά) ακροδέκτη. Είναι κατάλληλο για μετασχηματιστές και τροφοδοτικά που παρέχουν τάση DC, και χρησιμοποιείται ευρέως σε laptops, routers, συστήματα LED, και άλλες συσκευές. Η συμβατότητα μεταξύ του βύσματος και της υποδοχής είναι σημαντική για την ασφαλή και αξιόπιστη λειτουργία της συσκευής.

Τέλος, το **καλώδιο τύπου 30 AWG** [38] είναι ένα πολύ λεπτό, μονόκλωνο ή πολύκλωνο σύρμα με διάμετρο περίπου 0,255 mm. Λόγω της μικρής διατομής του, μπορεί να διαχειριστεί μέχρι περίπου 0,86 A και χρησιμοποιείται κυρίως σε κυκλώματα χαμηλής κατανάλωσης, όπως γραμμές σήματος, σύνδεση αισθητήρων ή debugging. Η ευελιξία και η λεπτή κατασκευή του το καθιστούν ιδανικό για εργασίες σε περιορισμένο χώρο ή για προσωρινές συνδέσεις σε πρωτότυπα κυκλώματα. Ακολουθεί συνοπτικός Πίνακας με τα τεχνικά χαρακτηριστικά (βλπ. Πίνακα 9).

Πίνακας 9:Συνοπτικός πίνακας τεχν. χαρακτηρ. εξαρτημάτων συνδεσιμότητας & ελέγχου

Εξάρτημα	Περιγραφή	Βήμα / Διάμετρος	Ρεύμα Λειτουργίας	Τάση Λειτουργίας	Σύνδεση	Ειδικά Χαρακτηριστικά
Pin Header	Γραμμικοί ακροδέκτες για σύνδεση πλακετών	2,54 mm	Έως 2-3 A	—	Μέσω τρύπας ή επιφανειακή	Διατίθενται σε αρσενικά/θηλυκά, κάθετα ή οριζόντια
JST XH Connector	Σύνδεσμος τύπου wire-to-board για σταθερή σύνδεση	2,54 mm	Έως 3 A	Έως 250 V	Καλώδιο σε πλακέτα	Μηχανικά ασφαλές κούμπωμα, 2–5 επαφές
On/Off Switch (SPST)	Διακόπτης για ενεργοποίηση /απενεργοποίηση	—	0,5–5 A	12–250 V AC/DC	Βιδωτή ή PCB	Τύποι: slide, toggle, rocker; δύο σταθερές καταστάσεις
Momentary Button	Στιγμιαίος διακόπτης επαφής για παλμική είσοδο	—	< 1 A	Έως 24 V	PCB ή panel mount	Επαναφέρεται αυτόματα μόλις αφεθεί
Καλώδιο 30 AWG	Λεπτό σύρμα για χαμηλά ρεύματα και σήματα	Ø 0,255 mm	Έως 0,86 A	—	Συγκόλληση ή crimp	Μόνωση PVC/σιλικόνης, ιδανικό για prototyping και debugging
DC Jack 5.5x2.5	Υποδοχή τροφοδοσίας DC για σύνδεση μετασχηματιστή	5,5 / 2,5 mm	Έως 5 A	Έως 30 V	PCB / καλώδιο	Κατάλληλο για laptops, routers, LEDs
Ποτενσιόμετρο 20K	Ρυθμιζόμενη αντίσταση για έλεγχο τάσης ή σήματος	—	< 0,5 A	Έως 200 V	PCB / καλώδιο	Περιστροφικό ή συρόμενο, ±10% ανοχή

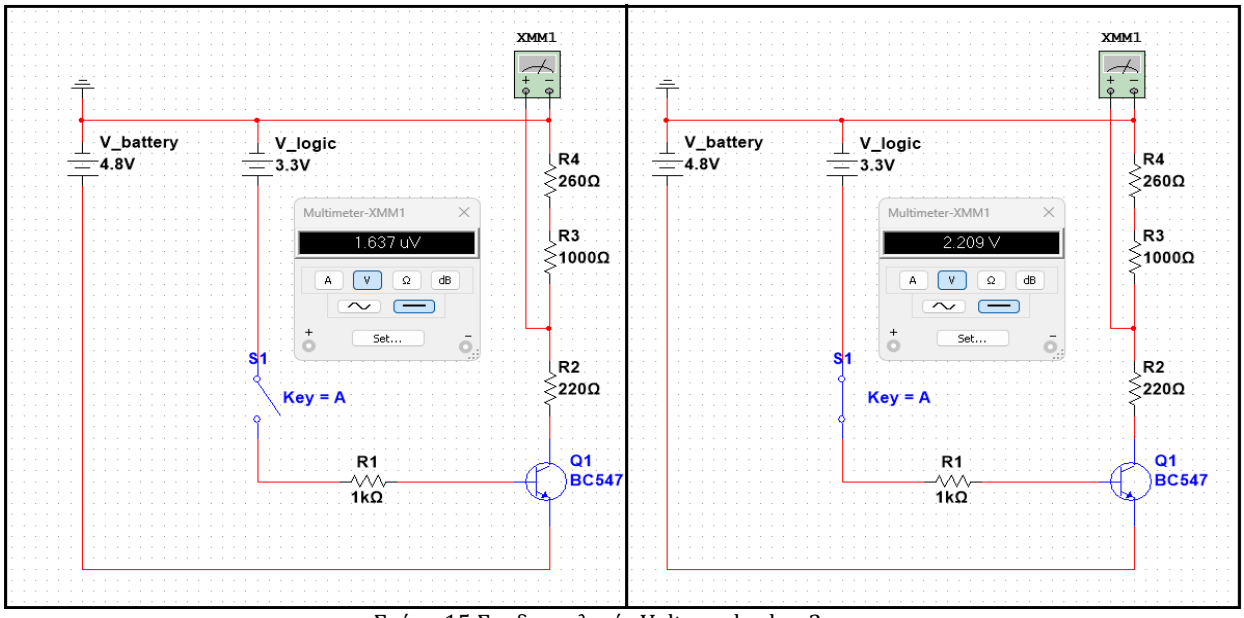
8. **Voltage Checker:** Ο ελεγκτής τάσης είναι ένα χειροποίητο ηλεκτρονικό εργαλείο που χρησιμοποιείται για τη μέτρηση της ηλεκτρικής τάσης (voltage) σε κυκλώματα, και συγκεκριμένα για τον έλεγχο της τάσης της μπαταρίας μιας συσκευής. Για την κατασκευή του κυκλώματος χρησιμοποιήθηκε ένα τρανζίστορ τύπου NPN BC547, δύο αντιστάσεις 1000Ω, μία αντίσταση 220Ω και μία αντίσταση 260Ω. Η συνδεσμολογία του κυκλώματος απεικονίζεται στην Εικόνα 24 και περιλαμβάνει τέσσερα καλώδια. Στο κόκκινο και το μαύρο καλώδιο εφαρμόζεται η τάση της μπαταρίας, ενώ μέσω του πράσινου καλωδίου μετριέται η τάση. Το μπλε καλώδιο χρησιμοποιείται για τον έλεγχο του τρανζίστορ.



Εικόνα 24:Συνδεσμολογία Voltage checker 1

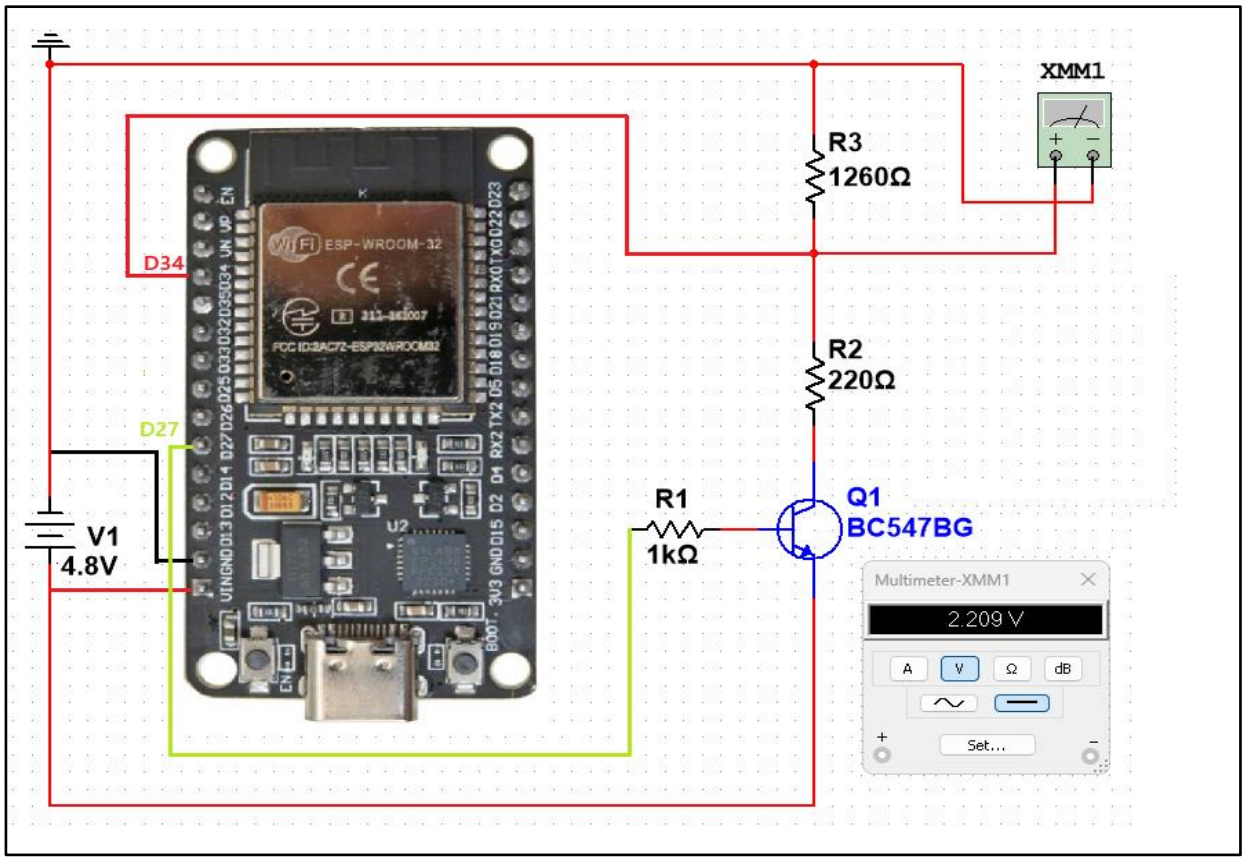
Συγκεκριμένα, το κύκλωμα χρησιμοποιεί ένα τρανζίστορ BC547B ως διακόπτη για να ελέγχει τη ροή ρεύματος προς ένα εξωτερικό φορτίο (R2,R3,R4) με βάση το σήμα που προέρχεται από το ESP32. Όταν το pin D27 του ESP32 δίνει υψηλό λογικό επίπεδο (3.3V), ρέει ρεύμα μέσω της αντίστασης R1 προς τη βάση του τρανζίστορ, το οποίο ενεργοποιείται και επιτρέπει τη ροή ρεύματος από τον συλλέκτη προς τον εκπομπό. Με αυτόν τον τρόπο, το φορτίο που αποτελείται από τις αντιστάσεις R2, R3 και R4 και είναι συνδεδεμένο στη γραμμή των 4.8V ενεργοποιείται, ενώ ταυτόχρονα στο pin D34 του ESP32 που έχει διαμορφωθεί ως αναλογική είσοδος, καταγράφει την τάση 2.209V. Η χρήση του διαιρέτη τάσης είναι απαραίτητη, καθώς το pin D34 μπορεί να διαβάσει μέγιστη τάση 3.3V και

συνεπώς πρέπει να προστατευτεί από υπερτάσεις. Η καταγραφή της συγκεκριμένης τιμής επιβεβαιώνει τη σωστή λειτουργία του κυκλώματος (βλπ. Σχήμα 15 και Εικόνα 25).



Σχήμα 15:Συνδεσμολογία Voltage checker 2

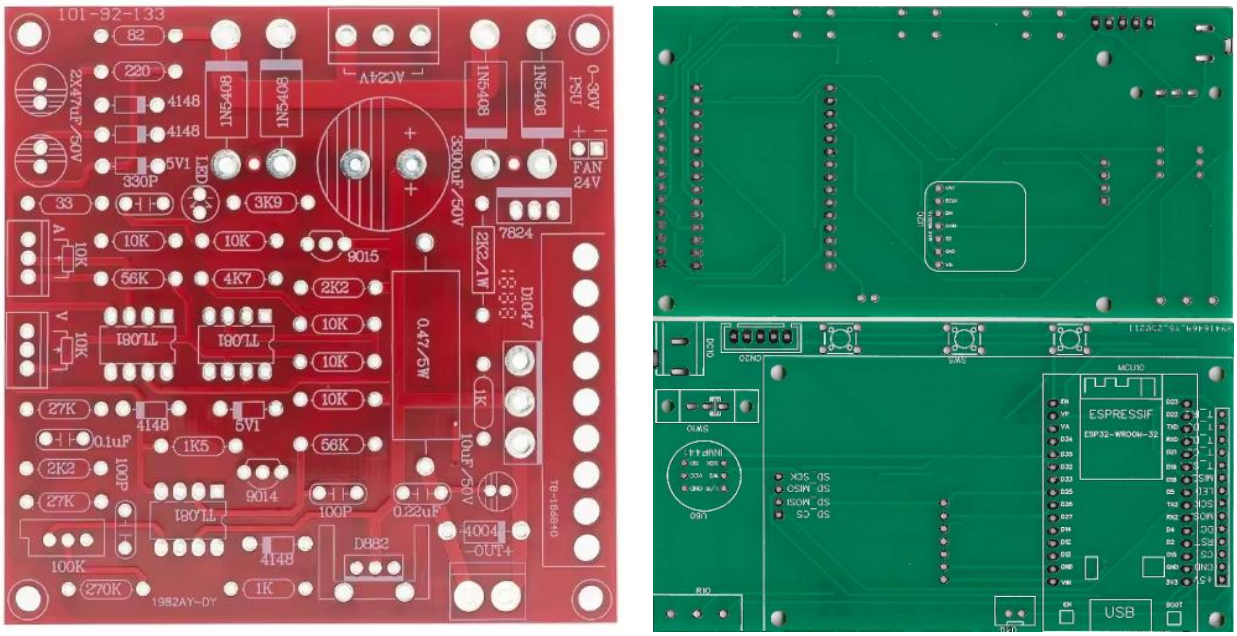
Αντίθετα, όταν το D27 βρίσκεται σε χαμηλό λογικό επίπεδο (0V), το τρανζίστορ παραμένει απενεργοποιημένο, με αποτέλεσμα να διακόπτεται η ροή ρεύματος προς το φορτίο και έτσι να μην έχουμε άσκοπη κατανάλωση ενεργείας της μπαταρίας.



Εικόνα 25:Συνδεσμολογία Voltage checker 3

Η απλή αυτή διάταξη επιτρέπει τη γρήγορη αξιολόγηση της κατάστασης της μπαταρίας μέσω της λειτουργίας του τρανζίστορ, το οποίο αντιδρά ανάλογα με την τάση που εφαρμόζεται στο κύκλωμα.

9. **PCB**: Το PCB, ή αλλιώς Τυπωμένο Κύκλωμα [39], είναι μια πλακέτα που χρησιμοποιείται για τη μηχανική στήριξη και τη σύνδεση ηλεκτρονικών εξαρτημάτων μέσω αγώγιμων διαδρομών που έχουν χαραχθεί σε ένα μονωτικό υπόστρωμα. Οι πλακέτες αυτές μπορούν να έχουν μία ή περισσότερες στρώσεις χαλκού και αποτελούν τη βάση για σχεδόν όλα τα ηλεκτρονικά κυκλώματα, από απλές συσκευές έως πολύπλοκα συστήματα. Τα εξαρτήματα τοποθετούνται πάνω στην πλακέτα είτε μέσω τεχνολογίας επιφανειακής στήριξης (SMT) είτε μέσω τρυπημένων οπών (through-hole) (βλπ. Εικόνα 26).



Εικόνα 26:Παραδείγματα πλακετών με τυπωμένο κύκλωμα (PCB)

10. **Μπαταρίες NiMH 1.2V AA και θήκη μπαταριών 4xAA**: Οι επαναφορτιζόμενες μπαταρίες τύπου AA 1.2V Ni-MH (Nickel-Metal Hydride) αποτελούν μια αξιόπιστη και ευρέως διαδεδομένη επιλογή για την παροχή ενέργειας σε φορητές ηλεκτρονικές συσκευές. Κάθε στοιχείο παρέχει ονομαστική τάση 1,2 Volt και χωρητικότητα που συνήθως κυμαίνεται μεταξύ 2000 και 3000 mAh, ανάλογα με τον κατασκευαστή και τα τεχνικά χαρακτηριστικά της εκάστοτε σειράς προϊόντος.

    Στο πλαίσιο της παρούσας διπλωματικής εργασίας επιλέχθηκαν επαναφορτιζόμενες μπαταρίες τύπου EXELL AA Ni-MH με χωρητικότητα 2200 mAh. Οι συγκεκριμένες μπαταρίες συνδυάζουν υψηλό βαθμό αξιοπιστίας, οικονομική αποδοτικότητα και περιβαλλοντική υπευθυνότητα, καθώς υποστηρίζουν μεγάλο αριθμό κύκλων φόρτισης και αποφόρτισης, περιορίζοντας έτσι τη χρήση μη επαναφορτιζόμενων στοιχείων.

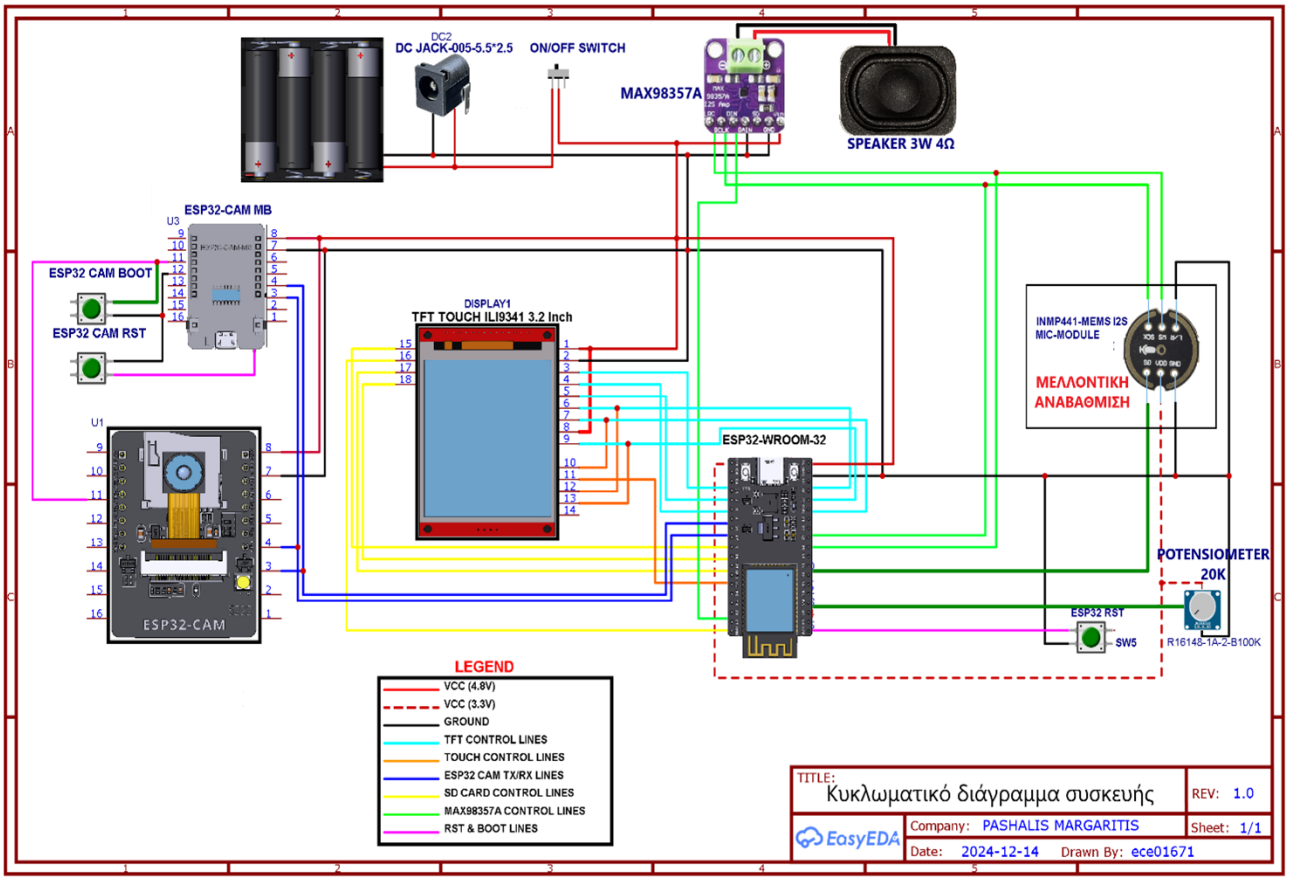
    Οι μπαταρίες τοποθετούνται σε ειδική πλαστική θήκη τύπου battery case, η οποία έχει σχεδιαστεί για τέσσερα στοιχεία τύπου AA και υποστηρίζει σειριακή σύνδεση, με αποτέλεσμα η συνολική τάση εξόδου να ανέρχεται στα 4,8V (1,2V × 4). Η θήκη διαθέτει ενσωματωμένο καλώδιο με βύσμα τύπου JST 2-Pin 2.54mm Male, προσφέροντας έτσι ασφαλή, σταθερή και εύκολη σύνδεση με την κύρια πλακέτα κυκλώματος (PCB).

    Η τεχνολογία Ni-MH προσφέρει πλεονεκτήματα όπως μεγαλύτερη διάρκεια ζωής σε σχέση με τις παραδοσιακές αλκαλικές, μειωμένο ποσοστό αυτοεκφόρτισης και σταθερή απόδοση ακόμα και σε εφαρμογές που απαιτούν συχνή φόρτιση. Συνολικά, η συγκεκριμένη λύση τροφοδοσίας κρίνεται ιδανική για φορητές ενσωματωμένες συσκευές, στις οποίες η αξιοπιστία, η ενεργειακή αυτονομία και η ευκολία συντήρησης αποτελούν βασικές απαιτήσεις (βλ. Εικόνα 27).

Εικόνα 27:4x 1.2V AA Nimh batteries και battery case 4xAA με JST 2Pin 2.54

3.3 Ηλεκτρονική - Κυκλωματική δομή

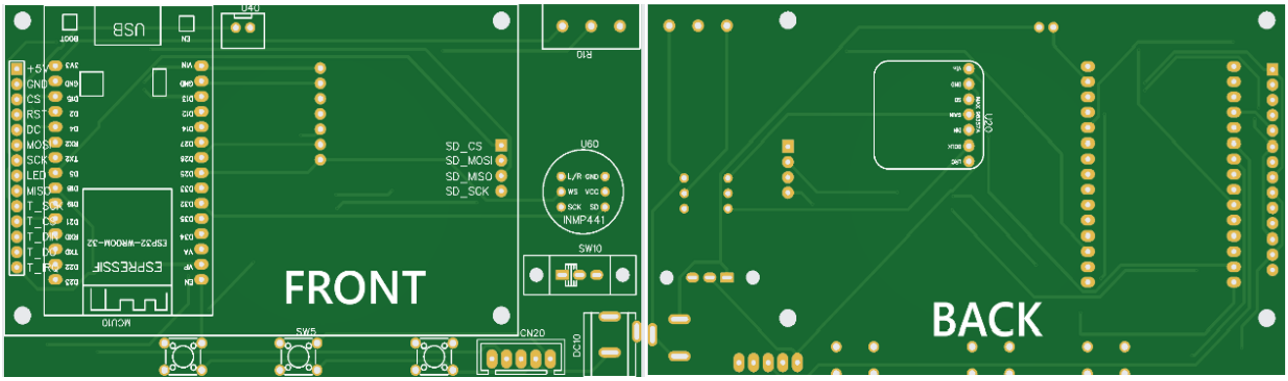
Η ηλεκτρονική και κυκλωματική δομή της συσκευής αποτελεί τη θεμελιώδη βάση πάνω στην οποία βασίζεται η λειτουργία του συστήματος. Μέσω της κατάλληλης επιλογής και διασύνδεσης ηλεκτρονικών εξαρτημάτων, διαμορφώνεται ένα ολοκληρωμένο κύκλωμα που εξασφαλίζει την αλληλεπίδραση μεταξύ των επιμέρους μονάδων, όπως μικροελεγκτές, αισθητήρες, οθόνες και διατάξεις εισόδου/εξόδου. Η σωστή κυκλωματική υλοποίηση επιτρέπει την αξιόπιστη μετάδοση σημάτων, την παροχή τροφοδοσίας και την εκτέλεση των προγραμματισμένων λειτουργιών, διασφαλίζοντας έτσι τη σταθερή και αποδοτική λειτουργία της συσκευής. Η παρουσίαση του κυκλώματος στο παρόν κεφάλαιο αποσκοπεί στην κατανόηση της εσωτερικής αρχιτεκτονικής και της αλληλεπίδρασης μεταξύ των εξαρτημάτων που το απαρτίζουν. Στο Σχήμα 16 παρουσιάζεται το συνολικό κύκλωμα της συσκευής, στο οποίο αποτυπώνονται αναλυτικά όλες οι ηλεκτρονικές συνδέσεις και τα επιμέρους εξαρτήματα που συμμετέχουν στη λειτουργία του συστήματος.



Σχήμα 16:Κυκλωματικό διάγραμμα συσκευής

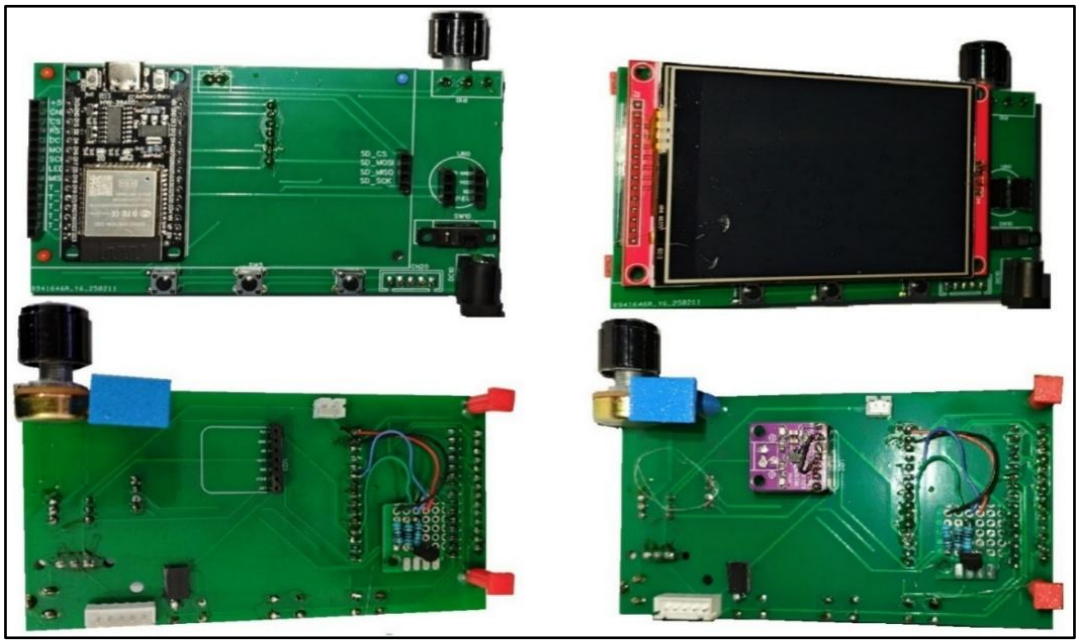
3.3.1 Πλακέτα τυπωμένου κυκλώματος

Η πλειονότητα των ηλεκτρονικών εξαρτημάτων τοποθετήθηκε σε πλακέτα τυπωμένου κυκλώματος (PCB), η οποία σχεδιάστηκε με τη βοήθεια του λογισμικού EasyEDA (βλπ. Εικόνα 28). Το συγκεκριμένο περιβάλλον επέτρεψε την αξιόπιστη αποτύπωση του ηλεκτρονικού διαγράμματος και τη βελτιστοποίηση της δρομολόγησης σημάτων, εξασφαλίζοντας σταθερή τάση τροφοδοσίας 4.8V, κοινό σύστημα γείωσης και ελαχιστοποίηση των ηλεκτρομαγνητικών παρεμβολών.



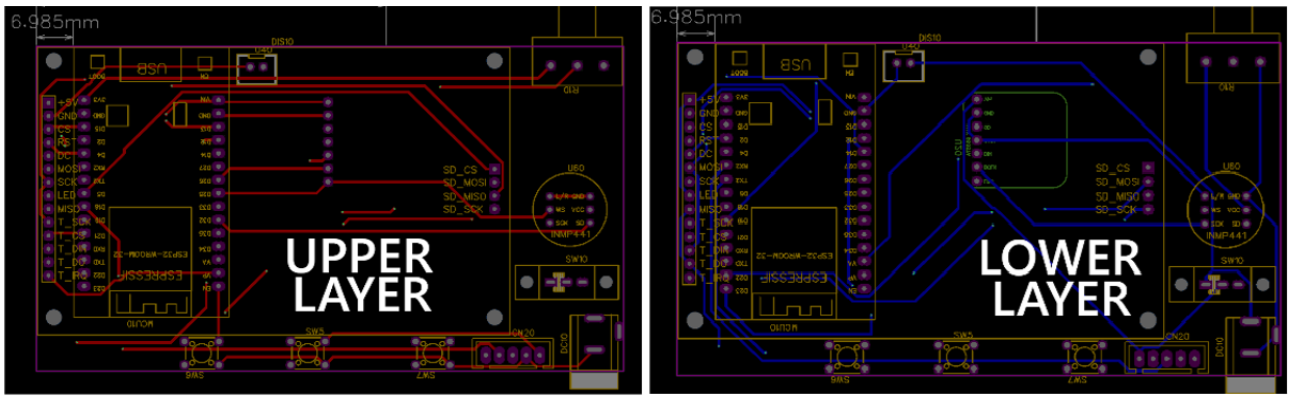
Εικόνα 28:Πλακέτα με το τυπωμένο κύκλωμα της συσκευής

Κατά τον σχεδιασμό δόθηκε ιδιαίτερη έμφαση στη λειτουργική εργονομία και στη διευκόλυνση της συντήρησης, με αποτέλεσμα μια συμπαγή και αξιόπιστη κατασκευή. Τα επιμέρους υποσυστήματα διασυνδέθηκαν μέσω pin headers και jumper καλωδίων, όπου αυτό κρίθηκε σκόπιμο, ώστε να εξασφαλίζεται εύκολη πρόσβαση για έλεγχο, επισκευή ή αντικατάσταση, χωρίς να απαιτείται η πλήρης αποσυναρμολόγηση της συσκευής. Παράλληλα, ορισμένα εξαρτήματα συγκολλήθηκαν απευθείας στην πλακέτα PCB, όπως το ποτενσιόμετρο, ο διακόπτης ενεργοποίησης (ON/OFF Switch) και το κύκλωμα Voltage Checker, εξασφαλίζοντας σταθερή και αξιόπιστη σύνδεση (βλ. Εικόνα 29). Η τοπολογία του κυκλώματος σχεδιάστηκε με επίκεντρο την κύρια μονάδα ESP32, η οποία λειτουργεί ως κεντρικός ελεγκτής και διαχειρίζεται την επικοινωνία με τις υπόλοιπες περιφερειακές μονάδες. Συγκεκριμένα, η κάμερα ESP32-CAM συνδέεται με κατάλληλες γραμμές ελέγχου, η έγχρωμη οθόνη αφής TFT ILI9341 μέσω του διαύλου SPI, ενώ η μετάδοση ήχου υλοποιείται μέσω του ψηφιακού ενισχυτή MAX98357A, ο οποίος επικοινωνεί με το ESP32 μέσω του διαύλου I2S.



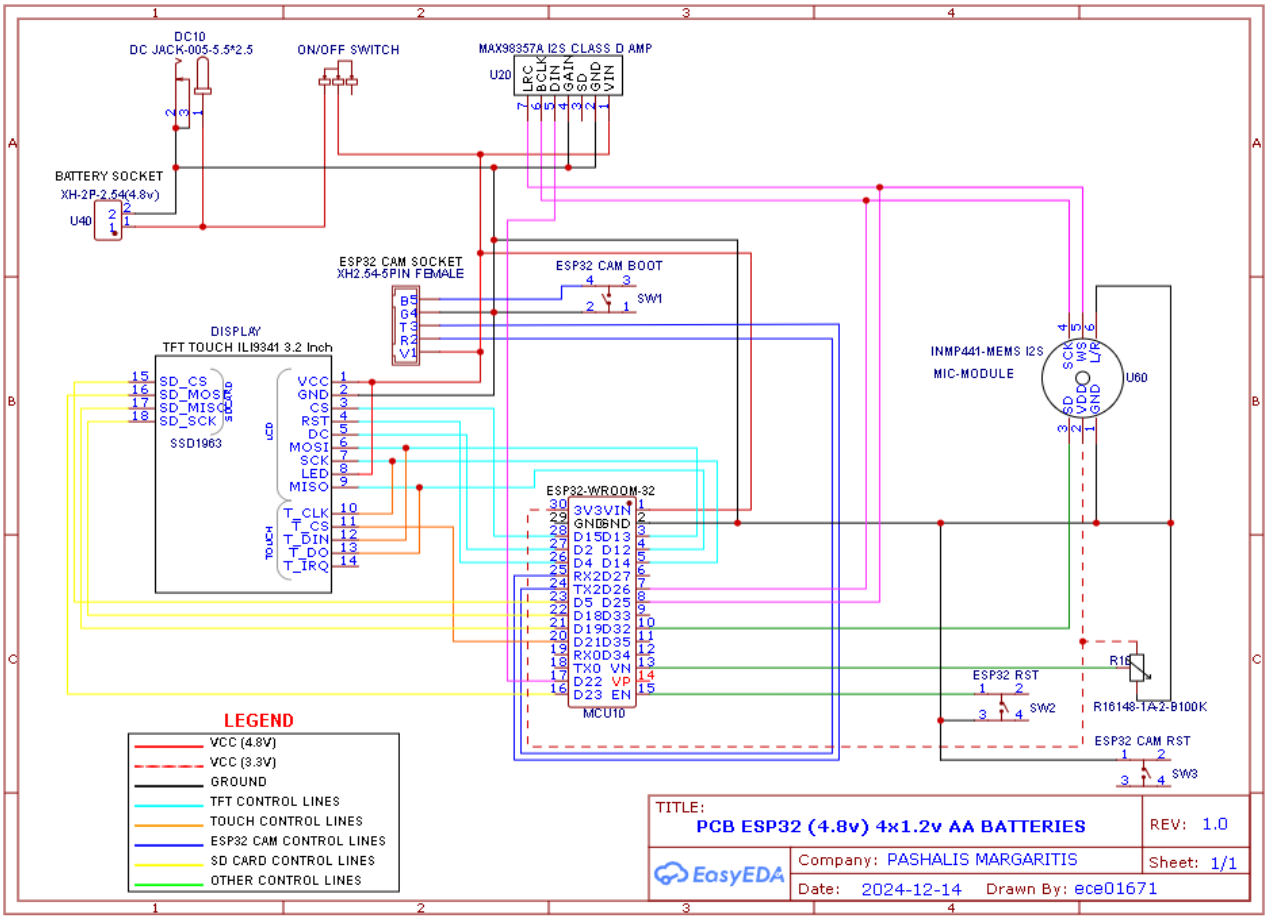
Εικόνα 29:Τοπολογία εξαρτημάτων στο άνω και κάτω μέρος της πλακέτας

Κατά τον σχεδιασμό, δόθηκε ιδιαίτερη έμφαση στη σωστή δρομολόγηση των σημάτων (signal routing), στην αποφυγή παρεμβολών μεταξύ γραμμών δεδομένων και στην ύπαρξη κοινής γείωσης (GND) για όλα τα εξαρτήματα. Οι γραμμές τροφοδοσίας σχεδιάστηκαν με αυξημένο πάχος, προκειμένου να αντέχουν το ρεύμα που απαιτείται από την οθόνη και τα υπόλοιπα περιφερειακά. Το κύκλωμα οργανώθηκε σε λογικά τμήματα, ώστε να διευκολύνεται τόσο η υλοποίηση όσο και η μελλοντική συντήρηση ή αναβάθμιση της συσκευής (βλπ. Εικόνα 30).



Εικόνα 30:Δρομολόγηση γραμμών διασύνδεσης της πλακέτας

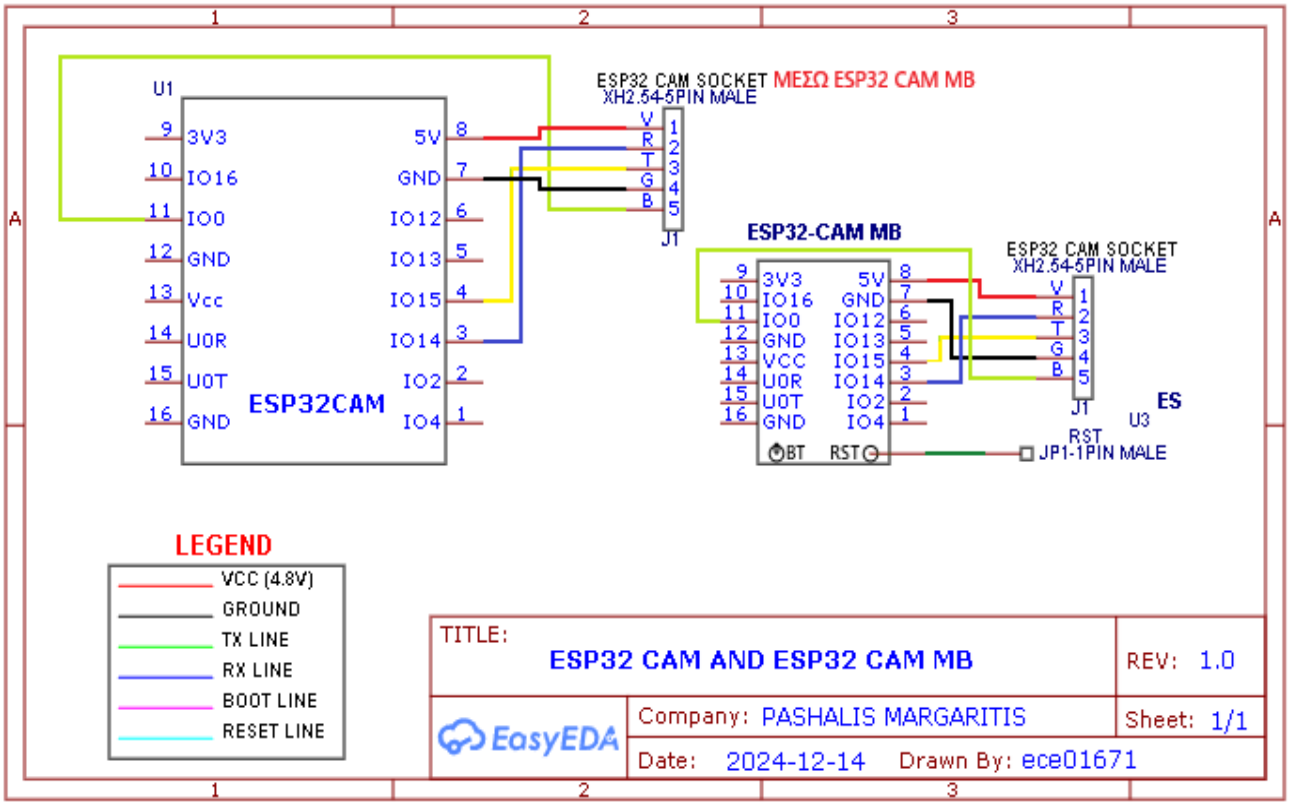
Η λεπτομερής συνδεσμολογία των επιμέρους εξαρτημάτων, καθώς και η αντιστοίχιση τους με τις κατάλληλες ακίδες (pins), παρουσιάζονται αναλυτικά στο ακόλουθο κυκλωματικό διάγραμμα (βλ. Σχήμα 17).



Σχήμα 17:Κυκλωματικό διάγραμμα εκτυπωμένης πλακέτας

Αφού ολοκληρώθηκε η περιγραφή της συνδεσμολογίας του PCB, είναι απαραίτητο να αναλυθεί η ακριβής συνδεσμολογία της κάμερας ESP32-CAM. Η διασύνδεση αυτής της μονάδας απαιτεί

συγκεκριμένες ηλεκτρικές και λογικές συνδέσεις, καθώς διαθέτει δικά της pins για τροφοδοσία και σειριακή επικοινωνία. Η σωστή ενσωμάτωσή της στο συνολικό κύκλωμα είναι καθοριστική για την ομαλή λειτουργία της συσκευής, καθώς αποτελεί τον βασικό αισθητήρα οπτικής εισόδου και επικοινωνεί άμεσα με την κεντρική μονάδα ESP32. Στο επόμενο τμήμα περιγράφεται αναλυτικά η συνδεσμολογία της ESP32-CAM με το κύριο ESP32 του κυκλώματος, καθώς και τα επιλεγμένα pins που εξασφαλίζουν σταθερή και αξιόπιστη επικοινωνία (βλπ. Σχήμα 18).



Σχήμα 18:Κυκλωματικό διάγραμμα ESP32 CAM και ESP32 CAM MB

Ο τελικός έλεγχος του σχεδίου έγινε με τη χρήση εργαλείων ηλεκτρονικού ελέγχου του EasyEDA, διασφαλίζοντας ότι δεν υπάρχουν λάθη στη σχεδίαση, όπως βραχυκυκλώματα ή ελλιπείς συνδέσεις. Το αποτέλεσμα ήταν ένα ολοκληρωμένο και λειτουργικό σχέδιο, έτοιμο για την παραγωγή της πλακέτας PCB και την εφαρμογή της στο φυσικό πρωτότυπο της συσκευής.

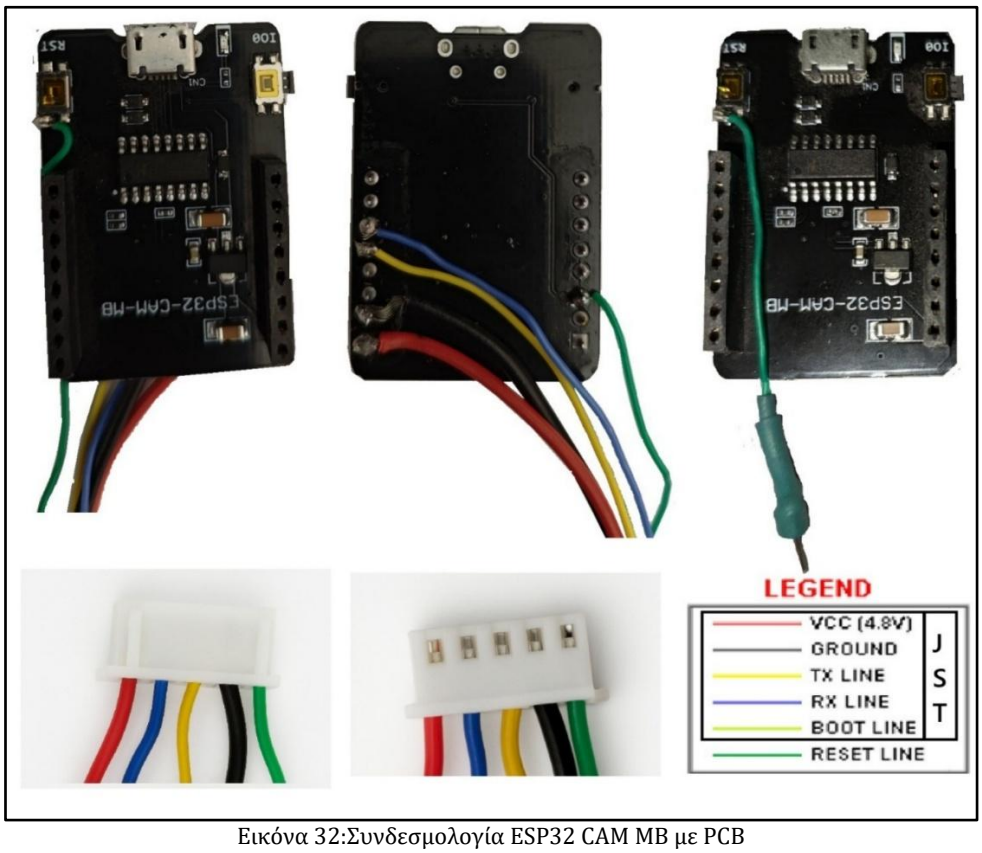
3.3.2 Συνδεσμολογία ESP32 CAM MB με ESP32 CAM

Η συνδεσιμότητα του ESP32-CAM με τη βάση προγραμματισμού ESP32-CAM-MB αποτελεί μια κρίσιμη και πρακτική διάταξη για την ανάπτυξη εφαρμογών IoT και αυτοματισμών. Η ESP32-CAM-MB είναι μια βοηθητική βάση που διευκολύνει τη σύνδεση και τον προγραμματισμό της μονάδας ESP32-CAM, παρέχοντας ενσωματωμένο USB to Serial μετατροπέα (CH340 ή CP2102) για απευθείας σύνδεση με υπολογιστή μέσω θύρας USB. Η μονάδα ESP32-CAM τοποθετείται απευθείας επάνω στη βάση μέσω αντίστοιχου socket, χωρίς την ανάγκη εξωτερικών jumpers ή πρόσθετων συνδέσεων. Μέσω των ακροδεκτών UART (TX/RX) και της παροχής ρεύματος 5V ή 3.3V, επιτυγχάνεται σταθερή επικοινωνία για upload κώδικα, debugging και τροφοδοσία του μικροελεγκτή. Επιπλέον, η βάση MB περιλαμβάνει κουμπί reset και auto-programming λειτουργία, επιτρέποντας την εύκολη μεταφόρτωση κώδικα από το Arduino IDE ή άλλο περιβάλλον ανάπτυξης, καθιστώντας τη λύση αυτή ιδανική για ταχύτερη και πιο αξιόπιστη ανάπτυξη πρωτοτύπων με ESP32-CAM (βλπ. Εικόνα 31).

Εικόνα 31:Συνδεσμολογία ESP32 CAM MB με ESP32 CAM

3.3.3 Συνδεσμολογία ESP32 CAM MB με το PCB

Η σύνδεση της μονάδας ESP32-CAM με το PCB πραγματοποιείται μέσω υποδοχής πέντε ακίδων και μίας επιπλέον μονής ακίδας, σχηματίζοντας ένα ολοκληρωμένο σύστημα συνδεσιμότητας που εξασφαλίζει τη σωστή λειτουργία και προγραμματισμό της μονάδας. Η υποδοχή των πέντε ακίδων περιλαμβάνει τα βασικά σήματα: VCC, GND, TX, RX και BOOT, ενώ η μονή ακίδα αντιστοιχεί στη γραμμή RESET. Μέσω των ακίδων VCC και GND, παρέχεται η απαιτούμενη τροφοδοσία στο κύκλωμα, ενώ οι γραμμές TX και RX χρησιμοποιούνται για τη σειριακή επικοινωνία με το άλλο ESP32. Η γραμμή BOOT, η οποία συνδέεται στο GPIO0, είναι απαραίτητη για την είσοδο της μονάδας σε λειτουργία προγραμματισμού (flash mode), ενώ η ακίδα RESET επιτρέπει την επανεκκίνηση της μονάδας είτε χειροκίνητα είτε μέσω λογισμικού. Η χρήση αυτών των ακίδων μέσω υποδοχών τύπου pin header διασφαλίζει μια αξιόπιστη και σταθερή ηλεκτρική σύνδεση, η οποία μπορεί να πραγματοποιείται και να επαναλαμβάνεται χωρίς την ανάγκη συγκόλλησης. Παράλληλα, διευκολύνεται η συντήρηση και ο έλεγχος του κυκλώματος, καθώς επιτρέπεται η εύκολη πρόσβαση στις ακίδες και η αποσύνδεση της μονάδας χωρίς επιπλέον εργαλεία ή μόνιμες επεμβάσεις (βλπ. Εικόνα 32).

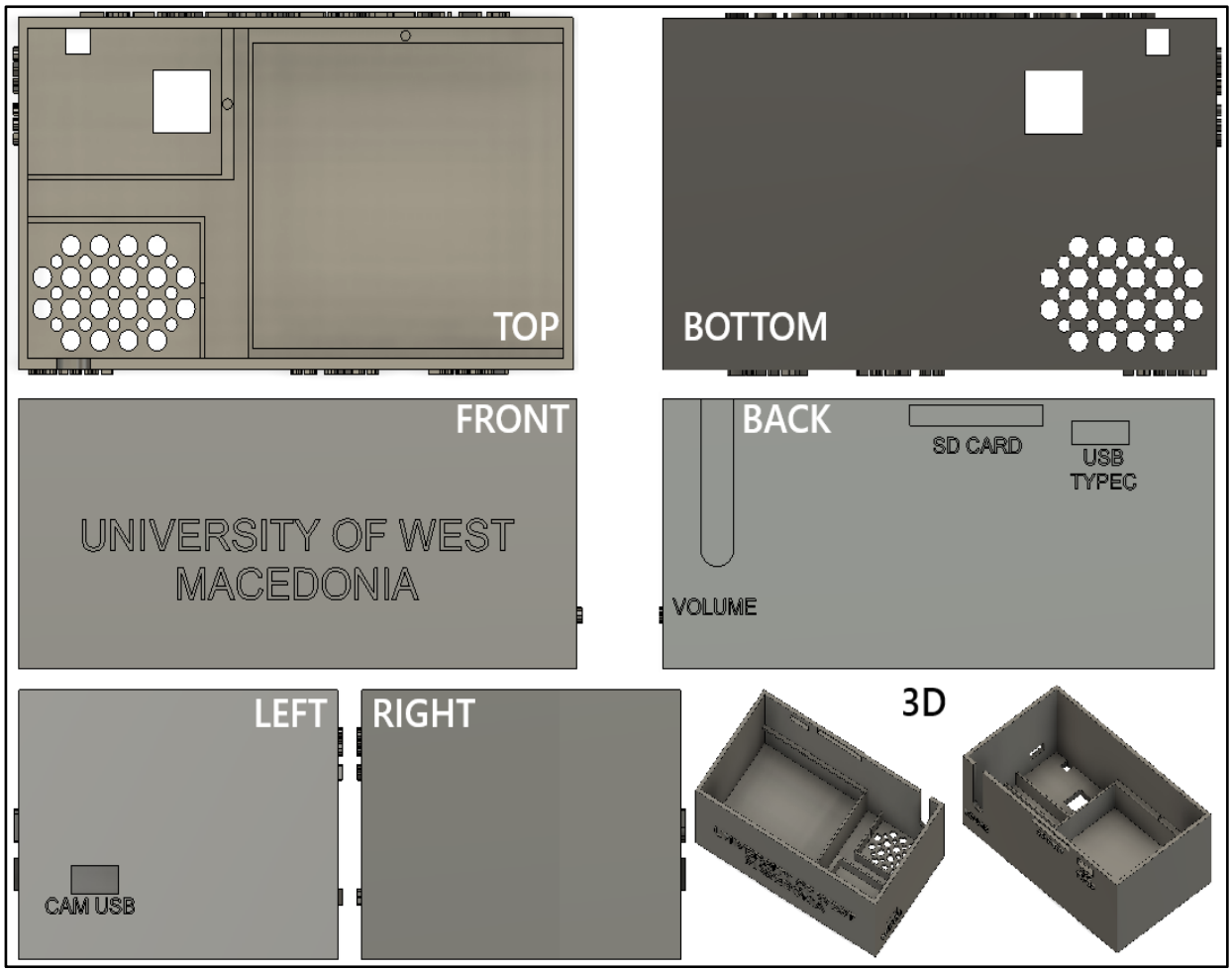


Εικόνα 32:Συνδεσμολογία ESP32 CAM MB με PCB

3.4 Φυσική δομή

Η φυσική δομή της συσκευής αποτελεί το σύνολο των υλικών και μηχανικών στοιχείων που περιβάλλουν και υποστηρίζουν την ηλεκτρονική της λειτουργία. Πρόκειται ουσιαστικά για το περίβλημα και τον εσωτερικό σκελετό που φιλοξενούν τα ηλεκτρονικά εξαρτήματα, προστατεύουν το κύκλωμα από μηχανικές καταπονήσεις και εξωτερικούς παράγοντες (όπως σκόνη, υγρασία ή χτυπήματα) και διευκολύνουν τη συνολική εργονομία και χρηστικότητα της συσκευής. Στην παρούσα εργασία, ο σχεδιασμός της φυσικής δομής πραγματοποιήθηκε με τη βοήθεια λογισμικού τρισδιάστατης μοντελοποίησης, ενώ η υλοποίησή της έγινε με εκτύπωση 3D χρησιμοποιώντας νήμα PLA. Ιδιαίτερη έμφαση δόθηκε στην εργονομία, την προσβασιμότητα των συνδέσεων και την εύκολη τοποθέτηση/αφαίρεση των εξαρτημάτων για λόγους συντήρησης ή αναβάθμισης.

- **PCB box: Το** PCB box αποτελεί ένα προστατευτικό και λειτουργικό περίβλημα που έχει σχεδιαστεί για να στεγάσει και να προστατεύει την πλακέτα κυκλώματος (PCB) και τα συνδεδεμένα εξαρτήματα του συστήματος. Κατασκευασμένο μέσω λογισμικού σχεδίασης 3D, όπως το Fusion 360, το κουτί αυτό είναι ειδικά προσαρμοσμένο στις διαστάσεις του PCB και στις ανάγκες του συγκεκριμένου project, περιλαμβάνοντας ανοίγματα και εγκοπές, όπως για Speaker, USB Type-C, υποδοχή κάρτας SD, CAM USB, Camera Lens αλλά και οπές αερισμού για θερμική διαχείριση (βλπ. Εικόνα 33).

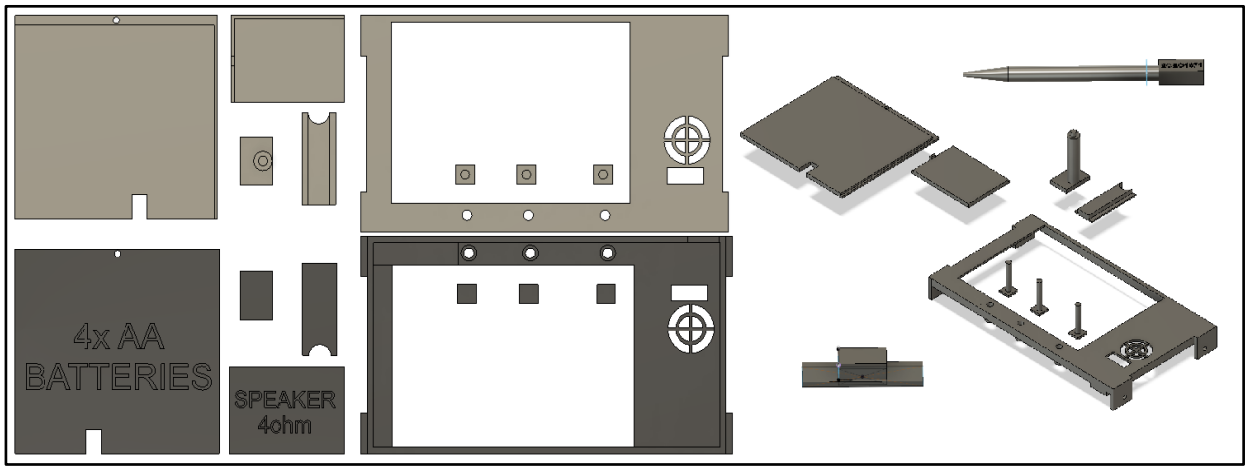


Εικόνα 33:3D μοντέλο Autodesk Fusion του PCB Box

Το φυσικό μοντέλο του PCB Box, που αποτελεί την υλοποιημένη εκδοχή του ψηφιακού Σχεδίου και προηγήθηκε στο λογισμικό Fusion, κατασκευάστηκε μέσω τρισδιάστατης εκτύπωσης με υλικό PLA χρησιμοποιώντας το slicer Creality Print (βλπ. Εικόνα 34).

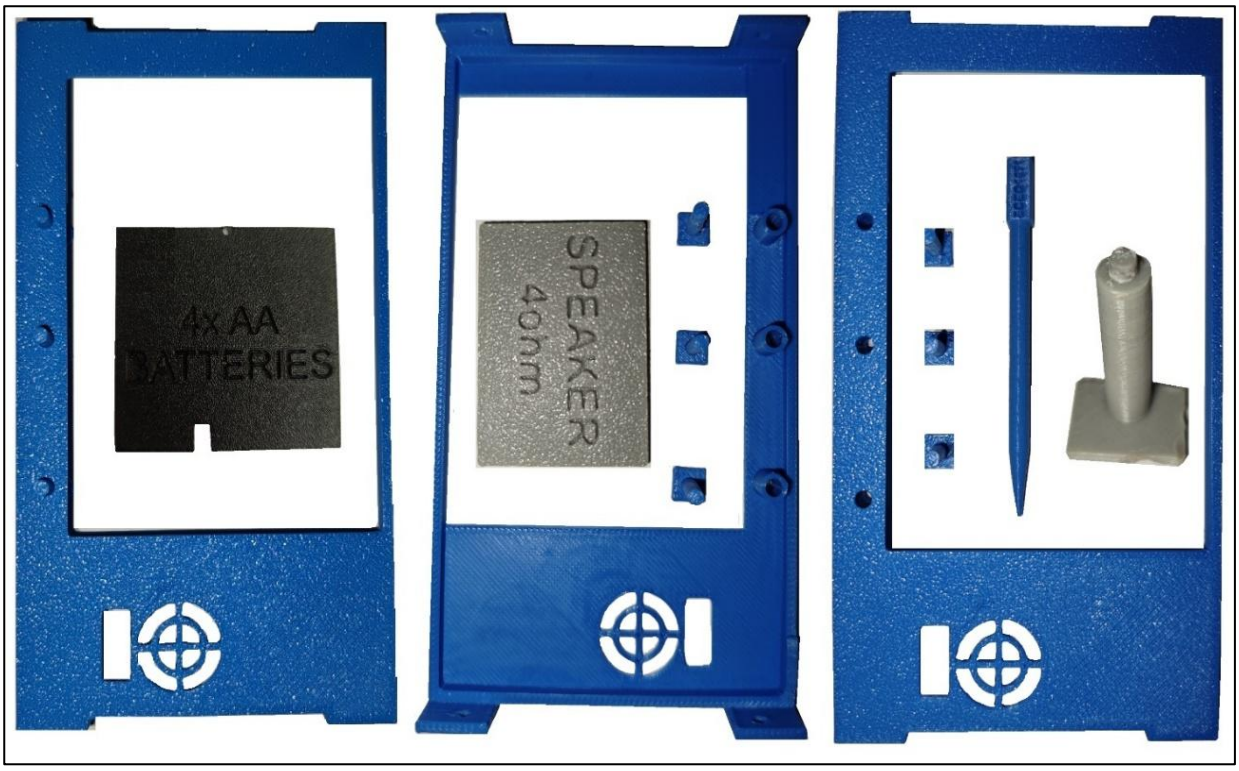
Εικόνα 34:Φυσικό μοντέλο του PCB Box

- **PCB box Lid:** Το καπάκι του PCB box αποτελεί αναπόσπαστο μέρος του συνολικού περιβλήματος και έχει σχεδιαστεί ώστε να προσφέρει εύκολη πρόσβαση στο εσωτερικό της κατασκευής, ενώ ταυτόχρονα διασφαλίζει την προστασία των ηλεκτρονικών εξαρτημάτων από εξωτερικούς παράγοντες όπως σκόνη, υγρασία και μηχανικές καταπονήσεις. Όπως και το υπόλοιπο κουτί, έτσι και το καπάκι σχεδιάζεται με το Fusion 360 και προσαρμόζεται με ακρίβεια στο σχήμα και τις διαστάσεις του κουτιού. Επίσης περιλαμβάνει επιπλέον ανοίγματα, κουμπιά, καθώς και μηχανισμούς ασφάλισης) για σταθερή και ασφαλή τοποθέτηση. Ομοίως σχεδιαστήκανε το τα καπάκια μπαταρίας και ηχείου, καθώς και ο στύλος αφής με τα κουμπιά ελέγχου και βάση PCB (βλπ. Εικόνα 35).



Εικόνα 35:3D Μοντέλο του PCB box lid, speaker lid, touch pen και batteries lid

Το φυσικό μοντέλο του PCB box Lid αποτελεί την, υλοποιημένη μορφή του αντίστοιχου ψηφιακού σχεδιασμού που δημιουργήθηκε στο Fusion 360. Όπως και το κυρίως σώμα του κουτιού, το καπάκι κατασκευάστηκε μέσω τρισδιάστατης εκτύπωσης με χρήση υλικού PLA, αξιοποιώντας τον slicer Creality Print. Η ακρίβεια της εκτύπωσης επέτρεψε την τέλεια εφαρμογή του καπακιού στο PCB box, διασφαλίζοντας τόσο τη μηχανική σταθερότητα όσο και την ευκολία πρόσβασης στο εσωτερικό της συσκευής. Ομοίως κατασκευάστηκαν τα καπάκια μπαταρίας και ηχείου, καθώς και ο στύλος αφής με τα κουμπιά ελέγχου και βάση PCB (βλπ. Εικόνα 36).



Εικόνα 36:Φυσικό Μοντέλο του PCB box lid, speaker lid, touch pen και batteries lid

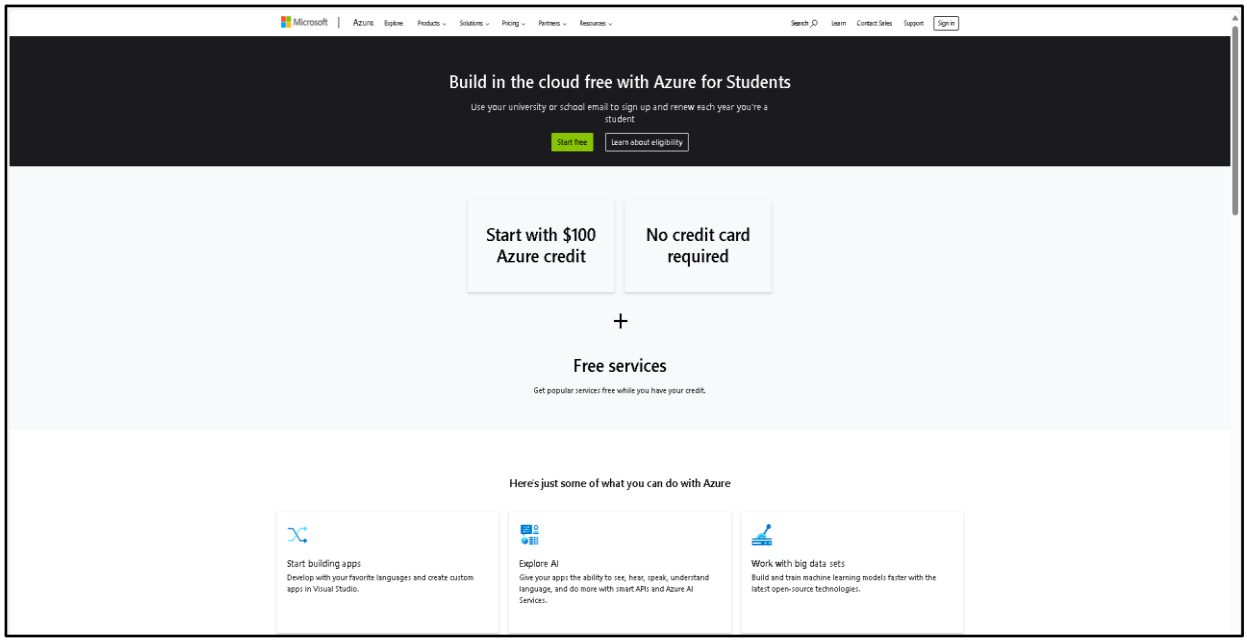
3.5 Λογισμική υποδομή

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, αξιοποιείται η υπολογιστική πλατφόρμα Microsoft Azure και συγκεκριμένα η υπηρεσία Computer Vision, με στόχο την αυτοματοποιημένη ανάλυση και επεξεργασία εικόνας. Η ενσωμάτωση της υπηρεσίας αυτής επιτρέπει την εξαγωγή χρήσιμων πληροφοριών από οπτικά δεδομένα, όπως ο εντοπισμός αντικειμένων, η αναγνώριση κειμένου (OCR), καθώς και η δημιουργία περιγραφών εικόνων με χρήση τεχνητής νοημοσύνης. Η επιλογή του Azure βασίστηκε στην ευκολία ενσωμάτωσης μέσω REST API, την υψηλή διαθεσιμότητα πόρων μέσω cloud και την υποστήριξη πολλαπλών γλωσσών προγραμματισμού.

3.5.1 Microsoft Azure

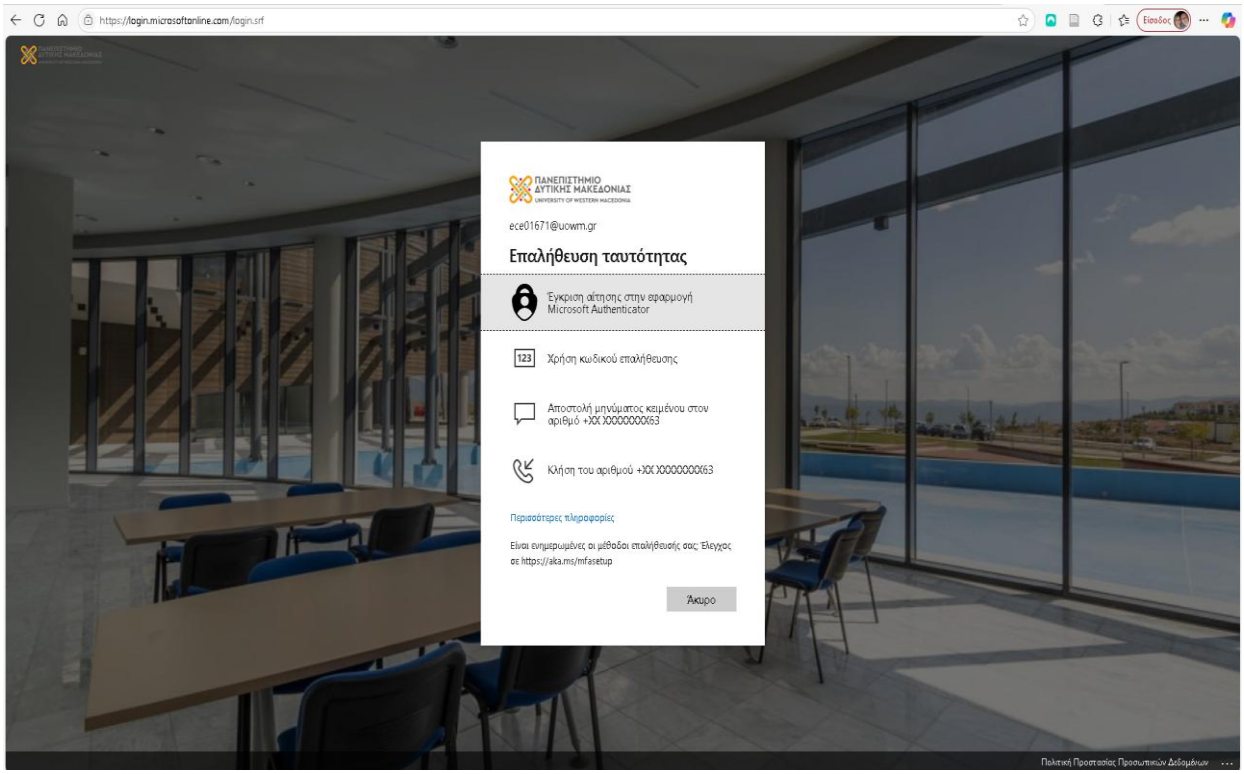
Αφού επιλέξαμε να χρησιμοποιήσουμε τις υπηρεσίες του Microsoft Azure και, πιο συγκεκριμένα, την υπηρεσία Computer Vision, το πρώτο βήμα ήταν η δημιουργία ενός λογαριασμού στην πλατφόρμα. Για τις ανάγκες της διπλωματικής εργασίας, έγινε χρήση του ιδρυματικού λογαριασμού, αξιοποιώντας τα εκπαιδευτικά προνόμια που προσφέρει η Microsoft μέσω του προγράμματος Azure for Students, το οποίο παρέχει δωρεάν πρόσβαση σε cloud υπηρεσίες χωρίς την απαίτηση χρήσης πιστωτικής κάρτας. Χρησιμοποιώντας το ιδρυματικό email (ece01671@uowm.gr), πραγματοποιήθηκε η εγγραφή στην πλατφόρμα και δημιουργήθηκε ένας εκπαιδευτικός λογαριασμός Azure. Ο λογαριασμός αυτός προσφέρει αρχικό πιστωτικό υπόλοιπο $100 καθώς και πρόσβαση σε βασικές υπηρεσίες, όπως το Azure AI Vision, όπου και θα κατασκευάσουμε ένα κλειδί Vision API, διευκολύνοντας έτσι την ανάπτυξη και υλοποίηση της εργασίας. Η διαδικασία δημιουργίας λογαριασμού στο Azure και η απόκτηση του Vision API Key περιγράφονται παρακάτω:

1. Μεταβαίνουμε στη σελίδα Azure for Students - https://azure.microsoft.com/en-us/free/students/ και πατάμε το Start Free (βλπ. Εικόνα 37).



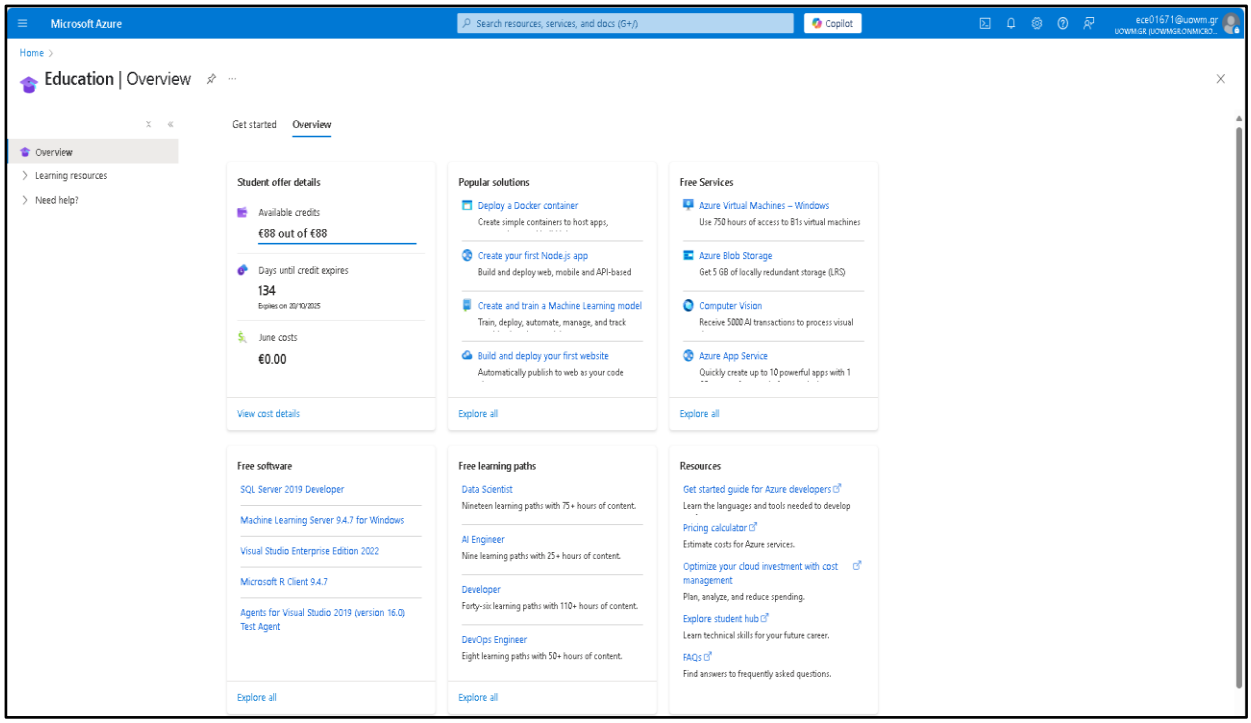
Εικόνα 37:Δημιουργία λογαριασμού Microsoft Azure – Βήμα 1
Πηγή: https://azure.microsoft.com/en-us/free/students?msockid=15c9b339131664441d39a6931261658f

2. Συνδεόμαστε με το ιδρυματικό σας email (π.χ. ece01671@uowm.gr). Συμπληρώνουμε τα προσωπικά μας στοιχεία και επαληθεύομε τον φοιτητικό μας λογαριασμό μέσω email ή με αριθμό κινητού τηλεφώνου (βλπ. Εικόνα 38).



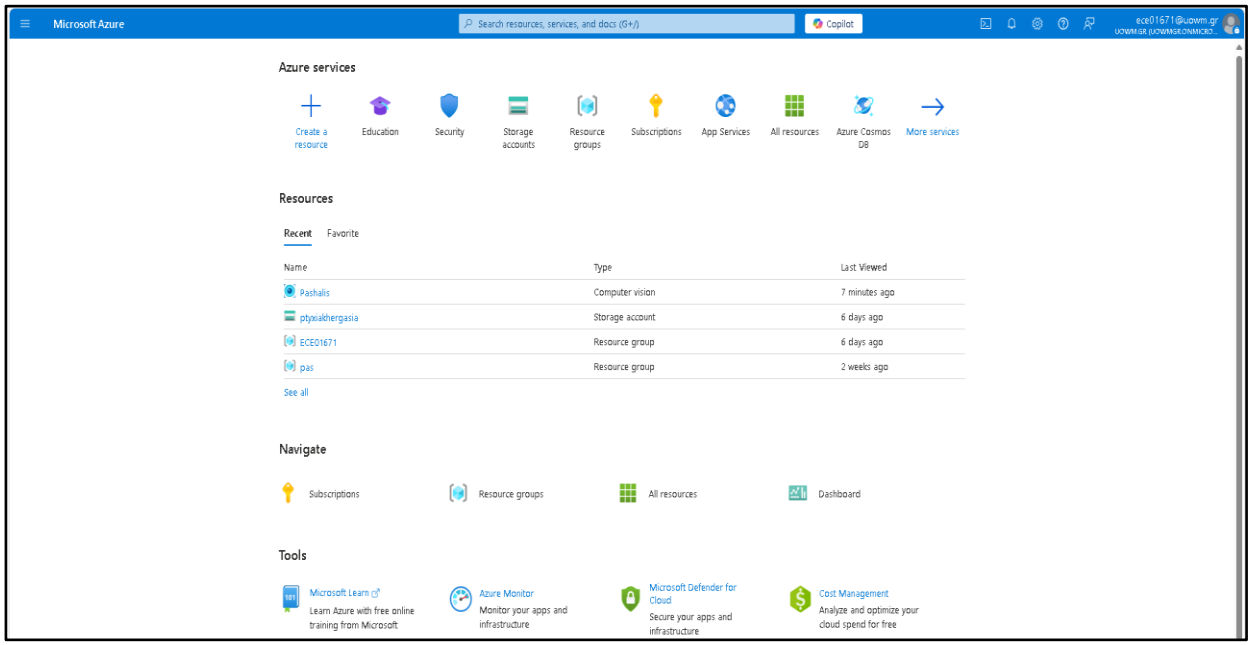
Εικόνα 38:Δημιουργία λογαριασμού Microsoft Azure – Βήμα 2

3. Αφού ολοκληρωθεί η εγγραφή, εμφανίζεται η αρχική οθόνη του λογαριασμού, η οποία παρέχει επισκόπηση των διαθέσιμων δωρεάν υπηρεσιών και λογισμικών. Ο λογαριασμός συνοδεύεται από ένα δωρεάν πιστωτικό ποσό (συνήθως $100) καθώς και πρόσβαση σε βασικές Azure υπηρεσίες για διάστημα 12 μηνών (βλ. Εικόνα 39).



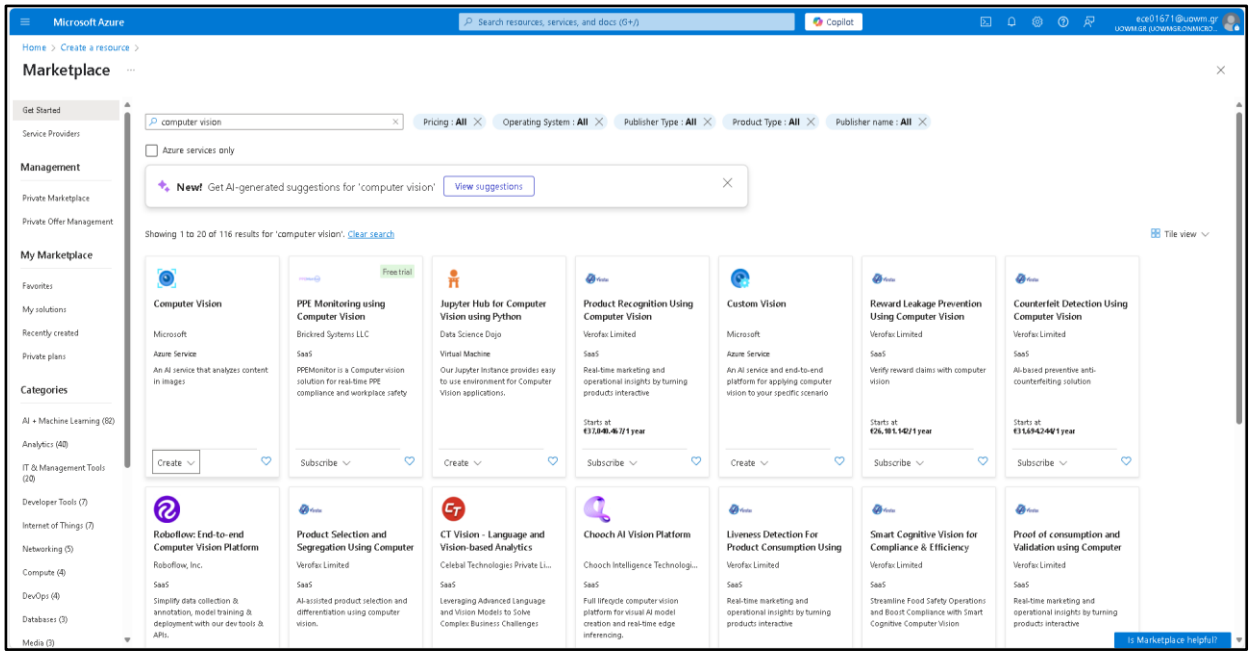
Εικόνα 39:Δημιουργία λογαριασμού Microsoft Azure – Βήμα 3

4. Αφού δημιουργηθεί επιτυχώς ο λογαριασμός, μεταβαίνουμε στη σελίδα του Azure Portal - https://portal.azure.com/ και συνδεόμαστε με τον φοιτητικό μας λογαριασμό (βλ. Εικόνα 40).



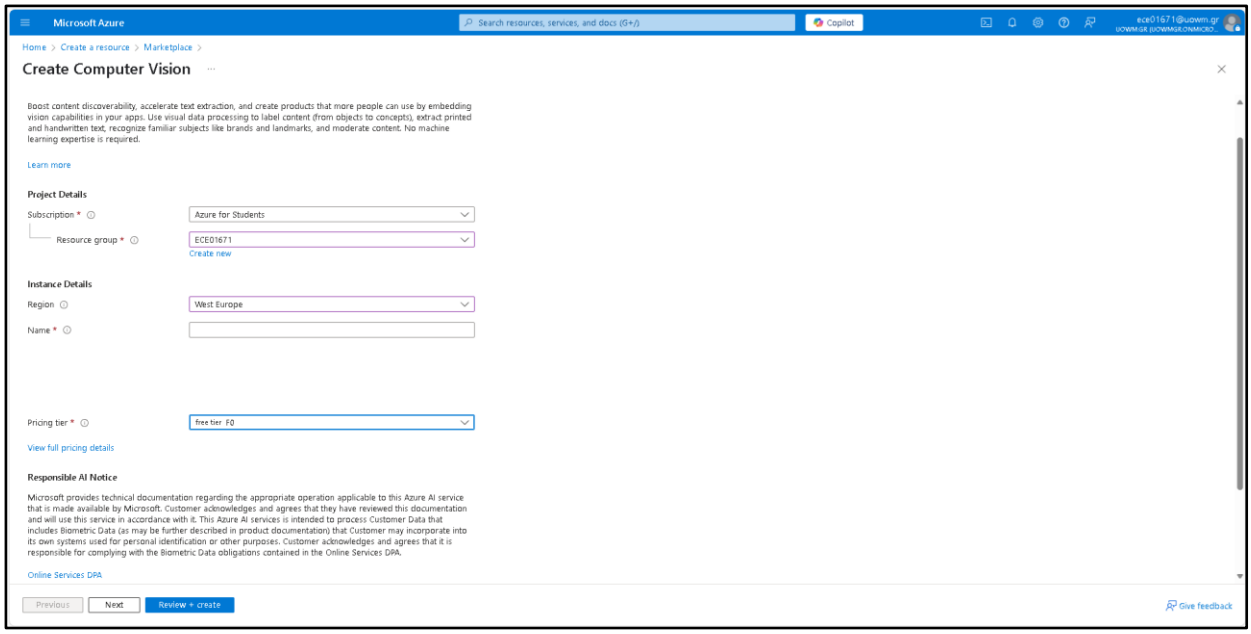
Εικόνα 40:Δημιουργία υπηρεσίας Azure Vision και λήψη API Key– Βήμα 1

5. Στην αρχική οθόνη του portal, επιλέγουμε "Create a resource" από το αριστερό μενού ή από το κουμπί στο κέντρο της σελίδας. Στο πεδίο αναζήτησης πληκτρολογούμε "Computer Vision" και επιλέγουμε την αντίστοιχη υπηρεσία (βλ. Εικόνα 41).



Εικόνα 41:Δημιουργία υπηρεσίας Azure Vision και λήψη API Key– Βήμα 2

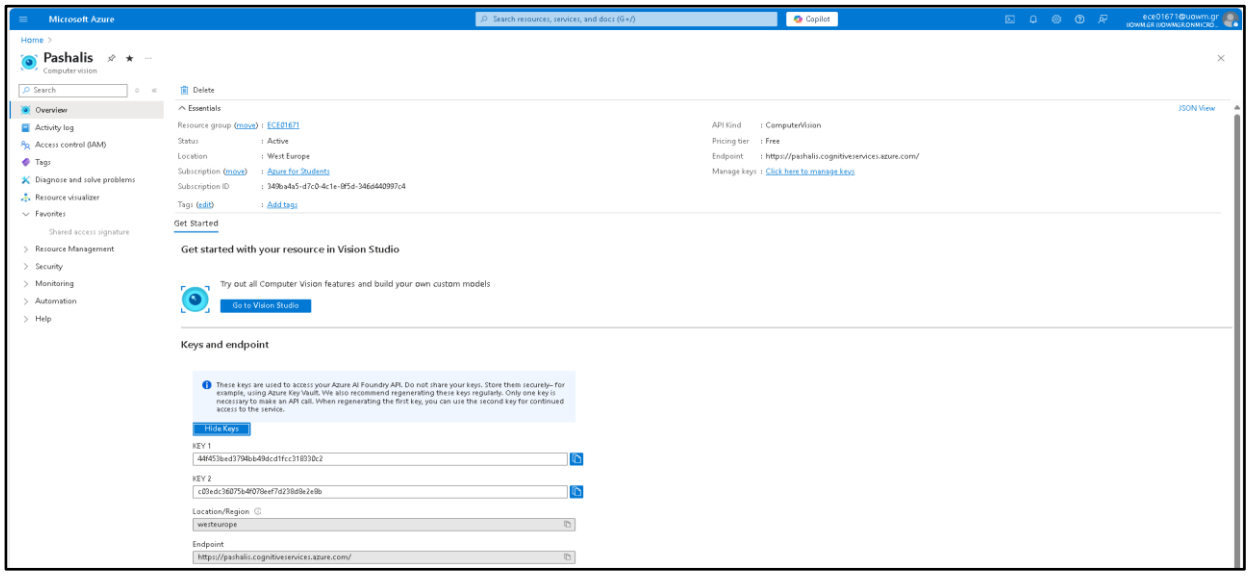
6. Στην οθόνη δημιουργίας της υπηρεσίας, αρχικά επιλέγουμε το πρόγραμμα Azure for Students από το πεδίο Subscription. Στη συνέχεια, δημιουργούμε ή επιλέγουμε ένα Resource group, στο παράδειγμα δημιουργούμε το ECE01671. Ως Region επιλέγουμε την περιοχή που θα φιλοξενήσει την υπηρεσία, όπως West Europe, ενώ στο πεδίο Name δίνουμε ένα όνομα για την υπηρεσία Computer Vision, όπως π.χ. Pashalis. Στην επιλογή Pricing tier διαλέγουμε το φοιτητικό πρόγραμμα Free tier (F0). Αφού συμπληρωθούν όλα τα απαραίτητα στοιχεία, μπορούμε να προχωρήσουμε με Review + Create και στη συνέχεια Create για να ολοκληρώσουμε τη διαδικασία δημιουργίας της υπηρεσίας (βλ. Εικόνα 42).



Εικόνα 42:Δημιουργία υπηρεσίας Azure Vision και λήψη API Key– Βήμα 3

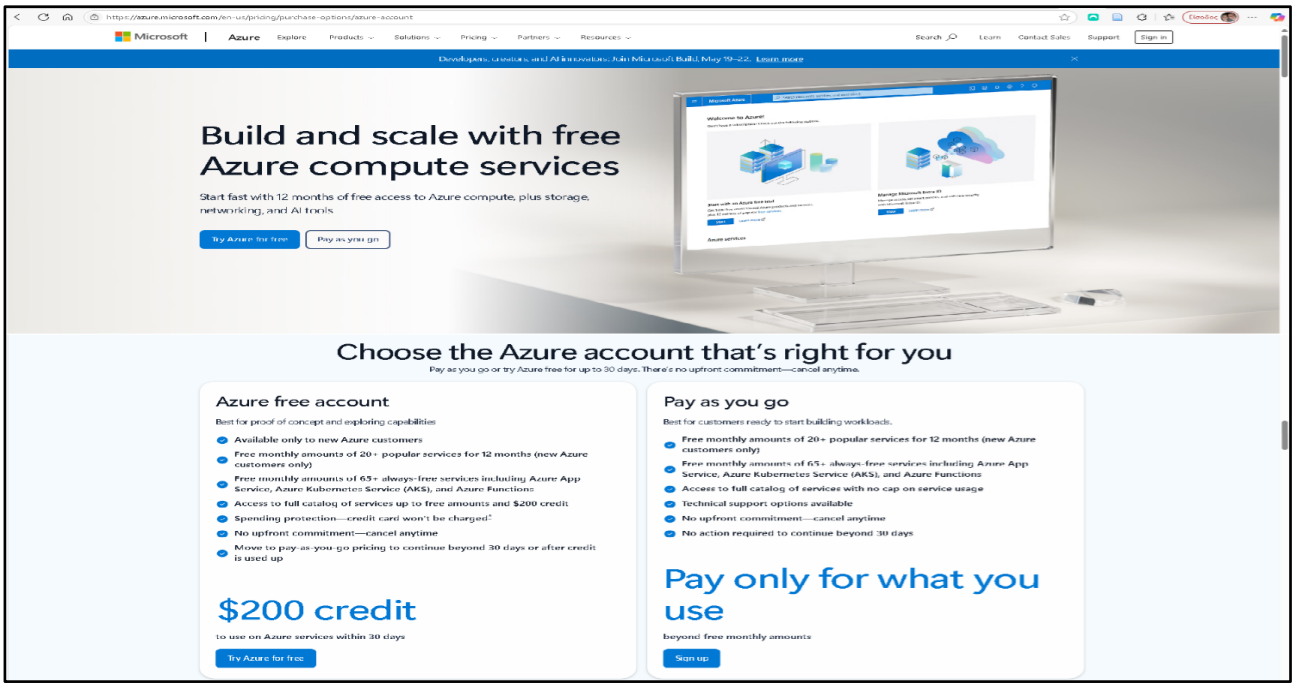
7. Μόλις ολοκληρωθεί η δημιουργία της υπηρεσίας, μεταβαίνουμε στη αρχική σελίδα και επιλέγουμε της υπηρεσίας Computer Vision (Pashalis) και στην ενότητα "Keys and Endpoint", θα δούμε δύο διαθέσιμα κλειδιά, το Key 1 και το Key 2, οποιοδήποτε από τα οποία μπορούμε να χρησιμοποιήσουμε για την αυθεντικοποίηση κατά την επικοινωνία με την υπηρεσία. Παράλληλα, παρέχεται και το Endpoint URL, το οποίο είναι απαραίτητο για

την κλήση της υπηρεσίας είτε μέσω REST API είτε μέσω κάποιας βιβλιοθήκης SDK (βλ. Εικόνα 43).



Εικόνα 43:Δημιουργία υπηρεσίας Azure Vision και λήψη API Key– Βήμα 4

Στην περίπτωση που δεν υπάρχει ιδρυματικός λογαριασμός, το Microsoft Azure εξακολουθεί να προσφέρει ένα ευέλικτο οικονομικό τρόπο χρήσης των υπηρεσιών της, κατάλληλο τόσο για αρχάριους όσο και για προχωρημένους χρήστες. Μέσω του Azure Free Account, οι νέοι χρήστες έχουν τη δυνατότητα να εξερευνήσουν τις δυνατότητες της πλατφόρμας χωρίς οικονομική δέσμευση, λαμβάνοντας δωρεάν πίστωση ύψους 200 δολαρίων για χρήση εντός 30 ημερών. Επιπλέον, παρέχεται δωρεάν μηνιαία πρόσβαση σε περισσότερες από 20 δημοφιλείς υπηρεσίες για διάστημα 12 μηνών, καθώς και πάνω από 65 υπηρεσίες που παραμένουν διαρκώς δωρεάν. Η διαδικασία ενεργοποίησης συνοδεύεται από προστασία χρέωσης, αφού η πιστωτική κάρτα δεν χρεώνεται αυτόματα, ενώ δεν απαιτείται καμία δέσμευση ή συμβόλαιο. Μετά την ολοκλήρωση των 30 ημερών ή την εξάντληση της πίστωσης, ο χρήστης μπορεί να συνεχίσει με το μοντέλο pay-as-you-go, πληρώνοντας μόνο για τις υπηρεσίες που χρησιμοποιεί (βλ. Εικόνα 44).



Εικόνα 44:Επιλογή λογαριασμού Azure

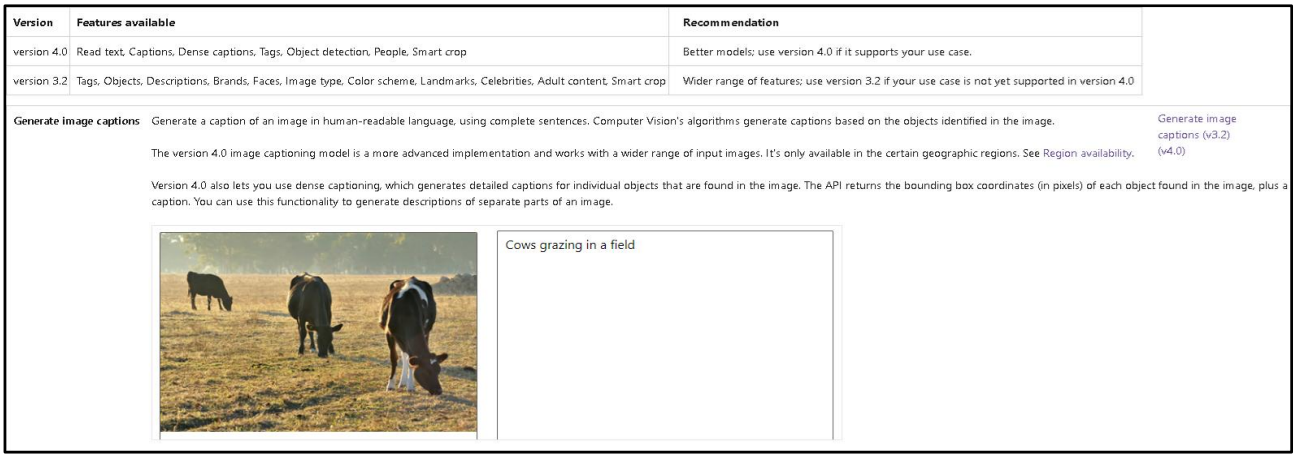
Πηγή: https://azure.microsoft.com/en-us/pricing/purchase-options/azure-account?msockid=15c9b339131664441d39

Επειδή η παρούσα διπλωματική εργασία υλοποιήθηκε σε γλώσσα προγραμματισμού C, η οποία δεν υποστηρίζεται άμεσα από το Microsoft Azure SDK, κρίθηκε σκόπιμη η χρήση του REST API της υπηρεσίας Azure Computer Vision. Μέσω αυτής της προσέγγισης, κατέστη δυνατή η αποστολή εικόνων για ανάλυση και η λήψη των αποτελεσμάτων σε μορφή JSON, διευκολύνοντας την επεξεργασία τους στο υπόλοιπο πρόγραμμα. Το REST API αποτελεί μια ευέλικτη και ανεξάρτητη λύση, καθώς βασίζεται στο πρωτόκολλο HTTP και μπορεί να χρησιμοποιηθεί από πληθώρα γλωσσών προγραμματισμού, χωρίς να απαιτείται επιπλέον λογισμικό ή σύνθετη εγκατάσταση. Το μόνο που απαιτείται είναι μια σύνδεση στο διαδίκτυο και τα αντίστοιχα διαπιστευτήρια, δηλαδή το API Key και το Endpoint που παρέχεται από την Azure.

Το **REST API** (Representational State Transfer Application Programming Interface) είναι ένα σύνολο από αρχές και κανόνες που επιτρέπουν σε διαφορετικά συστήματα λογισμικού να επικοινωνούν μεταξύ τους μέσω του διαδικτύου. Η επικοινωνία επιτυγχάνεται μέσω HTTP αιτημάτων, όπως:

- **GET** για ανάκτηση δεδομένων,
- **POST** για αποστολή δεδομένων,
- **PUT** για ενημέρωση,
- και **DELETE** για διαγραφή.

Για τον προγραμματισμό της συσκευής επιλέχθηκε η έκδοση 4.0 της υπηρεσίας Image Analysis του Microsoft Azure και συγκεκριμένα η Generate image captions. Η επιλογή της πιο πρόσφατης έκδοσης βασίστηκε στην αναβαθμισμένη λειτουργικότητα που προσφέρει, όπως η βελτιωμένη ακρίβεια στην αναγνώριση αντικειμένων, η ενισχυμένη δημιουργία περιγραφών εικόνας (captioning) με χρήση προηγμένων μοντέλων τεχνητής νοημοσύνης. Η έκδοση 4.0 διαθέτει αναλυτική τεκμηρίωση και σταθερή υποστήριξη μέσω REST API, γεγονός που διευκόλυνε την ενσωμάτωσή της στο λογισμικό της διπλωματικής εργασίας. Η χρήση της συγκεκριμένης έκδοσης εξασφάλισε πιο ακριβή και πλούσια ανάλυση εικόνας (βλ. Εικόνα 45).

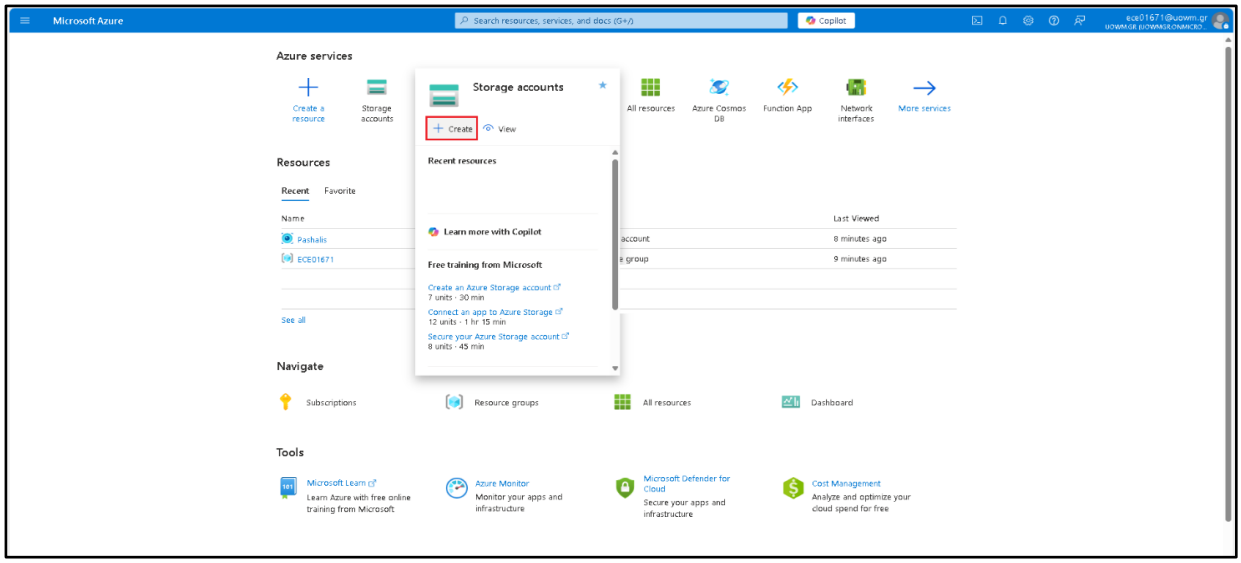


Εικόνα 45:Επιλογή έκδοσης 4.0 και υπηρεσίας Generate image captions
Πηγή: https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-image-analysis?tab=4-0

Η συσκευή, εκτός από τη δυνατότητα περιγραφής εικόνας, υποστηρίζει και **λήψη φωτογραφιών** μέσω της λειτουργίας CAMERA. Μετά τη λήψη, κάθε φωτογραφία αποθηκεύεται αυτόματα σε χώρο αποθήκευσης στο Azure Cloud. Για την υλοποίηση της αποθήκευσης δημιουργήθηκε ένα Azure Storage Account, στο οποίο προστέθηκαν δύο containers με σκοπό την οργάνωση και διαχείριση των αρχείων. Ο πρώτος container έχει την ονομασία camera-only και σε αυτόν αποθηκεύονται όλες οι φωτογραφίες που λαμβάνονται μέσω της λειτουργίας CAMERA. Η ονομασία κάθε αρχείου ακολουθεί τη μορφή YYYYMMDD_HHMMSS_captured_picture.jpg, όπως για παράδειγμα 20250602_141234_

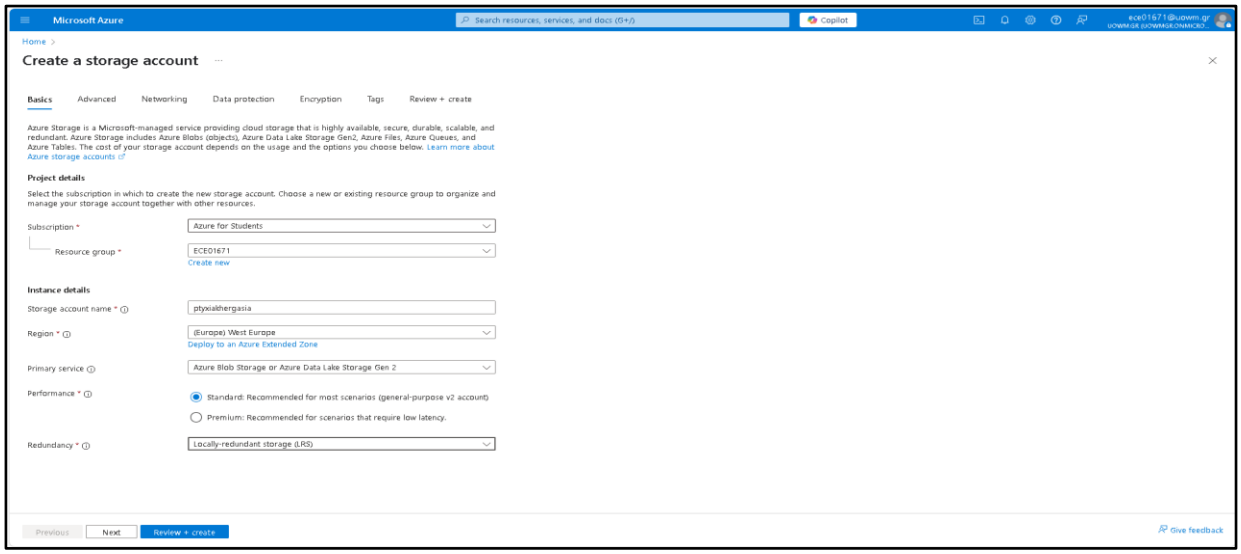
captured_picture.jpg. Η χρονοσφραγίδα (timestamp) προστίθεται αυτόματα κατά τη λήψη, ώστε να καταγράφεται με ακρίβεια η χρονική στιγμή λήψης της εικόνας. Ο δεύτερος container ονομάζεται computer-vision και χρησιμοποιείται για την αποθήκευση μίας μόνο φωτογραφίας κάθε φορά. Η φωτογραφία αυτή προορίζεται για ανάλυση μέσω της λειτουργίας Azure Computer Vision, επάνω στην οποία εφαρμόζεται η διαδικασία εξαγωγής περιγραφής εικόνας. Η διαδικασία δημιουργίας Storage Account περιγράφεται παρακάτω:

1. Η διαδικασία δημιουργίας ενός Storage Account στο Azure ξεκινά από την αρχική σελίδα του Azure Portal. Αρχικά, στην αρχική οθόνη του Azure περιβάλλοντος επιλέγουμε την ενότητα Storage accounts, η οποία βρίσκεται στο τμήμα "Azure services". Στη συνέχεια, στο αναδυόμενο παράθυρο που εμφανίζεται, πατάμε το κουμπί Create, προκειμένου να ξεκινήσει η διαδικασία δημιουργίας ενός νέου Storage Account (βλ. Εικόνα 46).



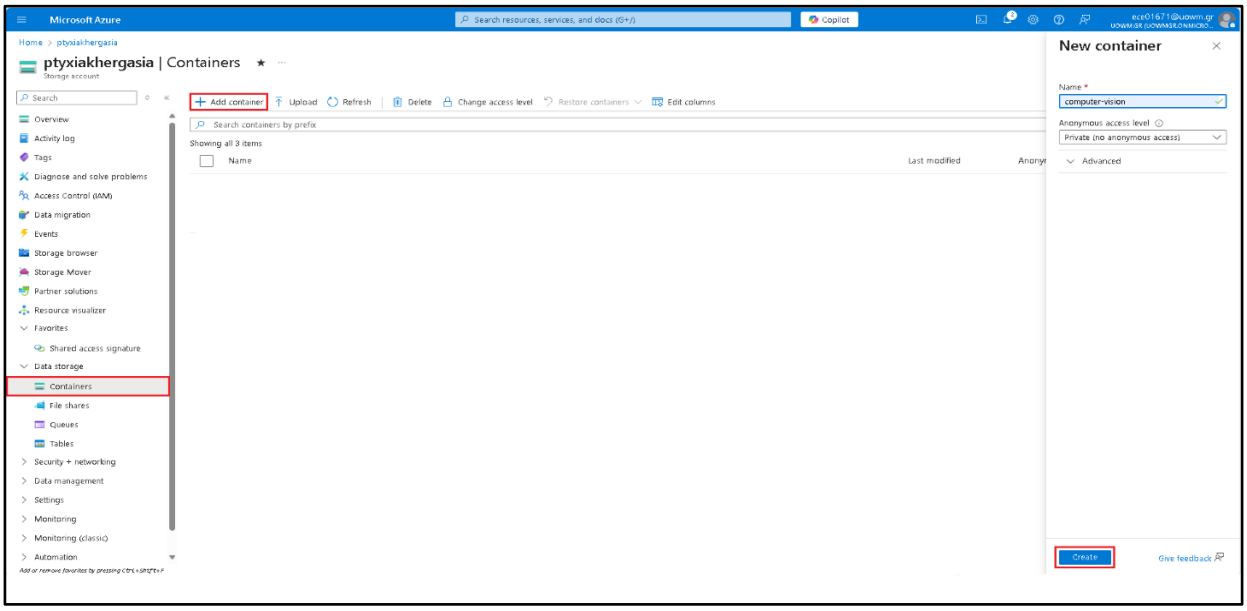
Εικόνα 46:Δημιουργία Storage account και Containers – Βήμα 1

2. Αυτή η ενέργεια μας μεταφέρει σε έναν οδηγό πολλαπλών βημάτων όπου συμπληρώνουμε τα απαιτούμενα πεδία, όπως το όνομα του λογαριασμού αποθήκευσης, την περιοχή (region), το group πόρων (resource group), τον τύπο απόδοσης (performance) και τις επιλογές πλεονασμού (redundancy). Αφού ολοκληρωθεί η συμπλήρωση των στοιχείων και γίνει επαλήθευση μέσω της επιλογής Review + Create, προχωρούμε στη δημιουργία του Storage Account πατώντας το κουμπί Rreview+Create (βλ. Εικόνα 47).



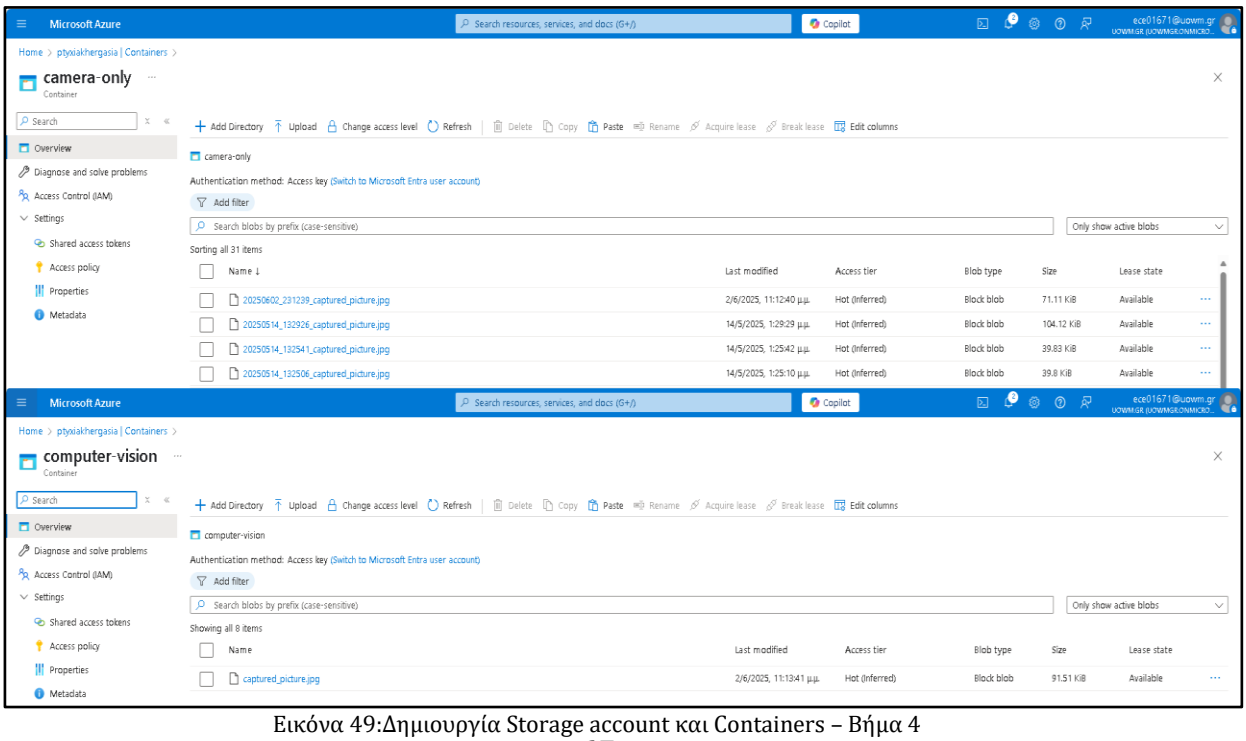
Εικόνα 47:Δημιουργία Storage account και Containers – Βήμα 2

3. Αφού ολοκληρωθεί η δημιουργία του Storage Account με επιτυχία, το επόμενο βήμα είναι η δημιουργία containers για την οργάνωση των αρχείων. Από την αρχική σελίδα του Storage Account, στην αριστερή πλαϊνή στήλη επιλέγουμε την ενότητα Containers. Στην κορυφή της σελίδας που εμφανίζεται, πατάμε το κουμπί + Add container για να ξεκινήσει η διαδικασία δημιουργίας νέου container. Στο δεξί τμήμα της οθόνης εμφανίζεται η φόρμα ρύθμισης του νέου container. Εκεί στο πεδίο Name, πληκτρολογούμε το όνομα του container camera-only, στο πεδίο Anonymous access level, επιλέγουμε το επίπεδο πρόσβασης. Για λόγους ασφαλείας συνιστάται η επιλογή Private (no anonymous access). Αφού συμπληρωθούν τα στοιχεία, επιλέγουμε Create στο κάτω μέρος της φόρμας. Ο νέος container δημιουργείται και εμφανίζεται στη λίστα. Η ίδια διαδικασία ακολουθείται και για τη δημιουργία του δεύτερου container, με όνομα computer-vision (βλ. Εικόνα 48).



Εικόνα 48:Δημιουργία Storage account και Containers – Βήμα 3

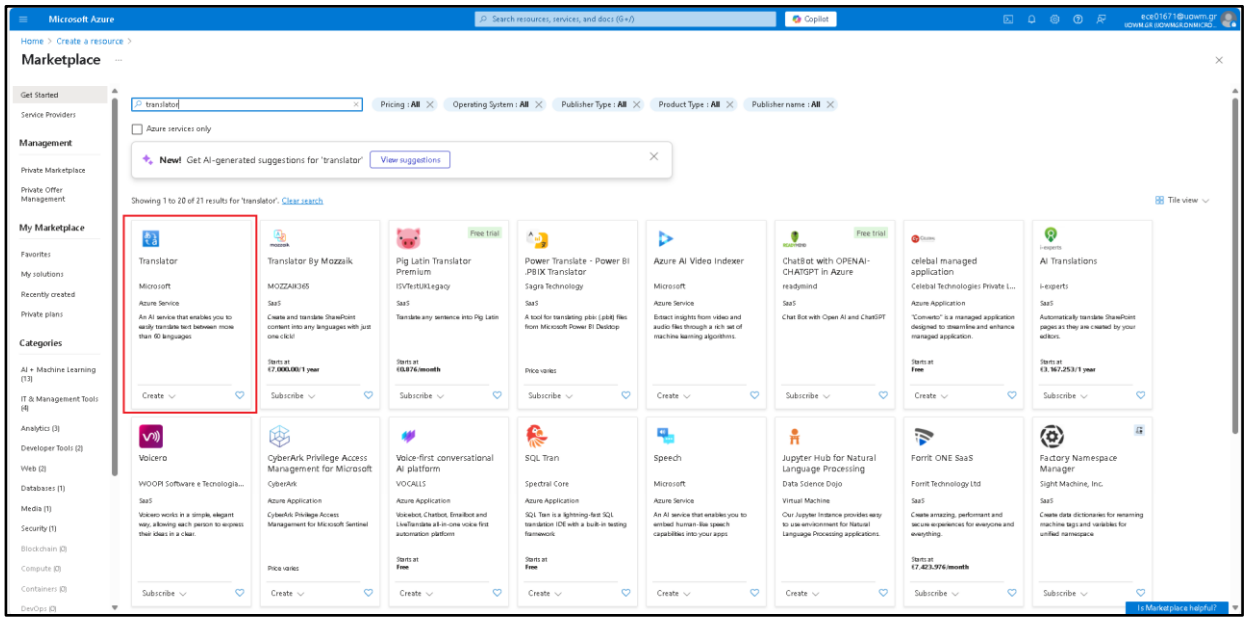
4. Η παρακάτω εικόνα παρουσιάζει τη δομή και το περιεχόμενο των δύο containers που έχουν δημιουργηθεί εντός του Azure Storage Account με όνομα ptyxiakhergasia (βλ. Εικόνα 49).



Εικόνα 49:Δημιουργία Storage account και Containers – Βήμα 4

Για την υλοποίηση της λειτουργίας αυτόματης μετάφρασης κειμένου, αξιοποιήθηκε η υπηρεσία **Azure Translator**, η οποία ανήκει στο σύνολο των υπηρεσιών τεχνητής νοημοσύνης της πλατφόρμας Azure Cognitive Services. Η δημιουργία της υπηρεσίας πραγματοποιείται με τα εξής βασικά βήματα:

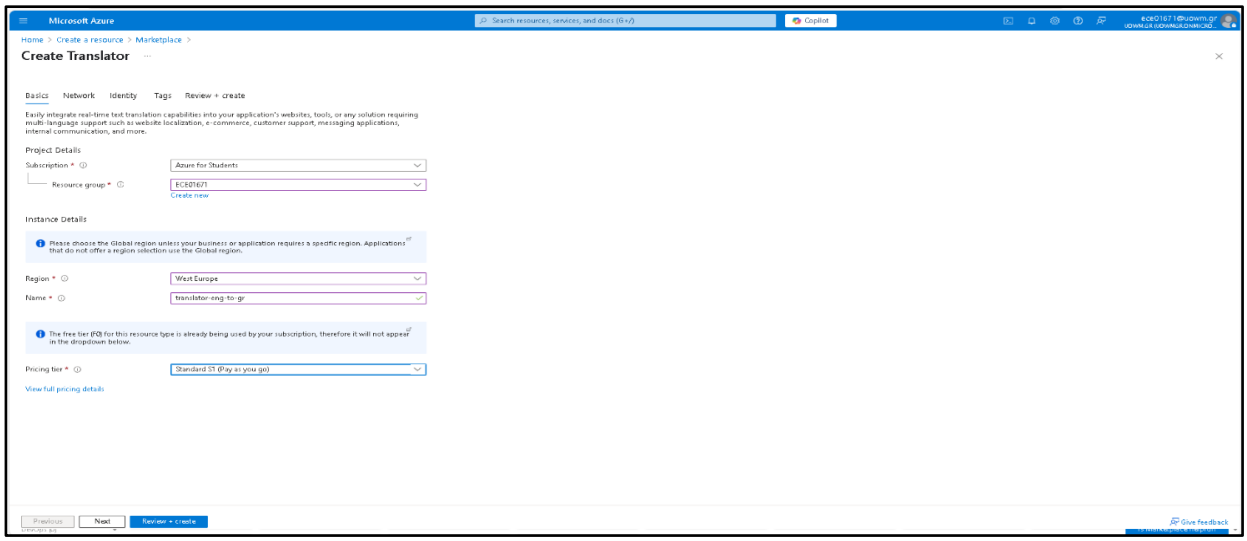
1. Αρχικά, απαιτείται η πρόσβαση στον λογαριασμό Azure του χρήστη και η επιλογή της ενέργειας **"Create a resource"** (Δημιουργία πόρου). Στο πεδίο αναζήτησης εισάγεται ο όρος **"Translator"** και επιλέγεται ο αντίστοιχος πόρος που παρέχεται από τη Microsoft. (βλ. Εικόνα 50).



Εικόνα 50:Δημιουργία Translator – Βήμα 1

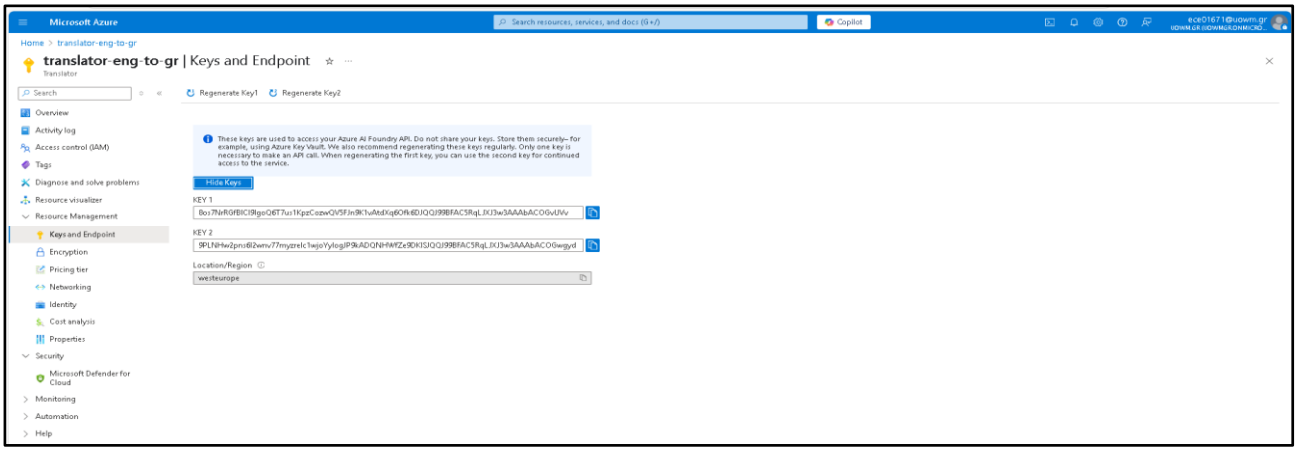
2. Αφού επιλεγεί ο πόρος, ο χρήστης καλείται να συμπληρώσει τις απαραίτητες πληροφορίες, όπως:
   - **Συνδρομή (Subscription)** και **ομάδα πόρων (Resource Group)**,
   - **Όνομα του πόρου**,
   - **Περιοχή (Region)**, η οποία στη συγκεκριμένη υλοποίηση είναι το **West Europe**,
   - **Κατηγορία τιμολόγησης (Pricing Tier)**, όπου για δοκιμαστικούς σκοπούς μπορεί να επιλεγεί το δωρεάν πλάνο (F0).

   Αφού ολοκληρωθεί η διαμόρφωση, ο πόρος δημιουργείται και είναι άμεσα διαθέσιμος για χρήση(βλ. Εικόνα 51).



Εικόνα 51:Δημιουργία Translator – Βήμα 2

Στη συνέχεια, από τη σελίδα του πόρου, ο χρήστης μπορεί να αποκτήσει το κλειδί πρόσβασης (Subscription Key) και τη διεύθυνση πρόσβασης (Endpoint URL), τα οποία είναι απαραίτητα για την αποστολή αιτημάτων HTTP προς την υπηρεσία. Η υπηρεσία Azure Translator επιτρέπει την αποστολή αιτημάτων μετάφρασης μέσω του τελικού σημείου (endpoint) https://api.cognitive.microsofttranslator.com/, με τη χρήση του πρωτοκόλλου REST και φορμάτ JSON. Η λειτουργία αξιοποιείται στη συγκεκριμένη υλοποίηση για τη μετάφραση των αποτελεσμάτων του Azure Computer Vision API από τα Αγγλικά στα Ελληνικά, με σκοπό τη βελτίωση της προσβασιμότητας των πληροφοριών από ελληνόφωνους χρήστες (βλ. Εικόνα 52).



Εικόνα 52:Δημιουργία Translator – Βήμα 3

Η ενσωμάτωση της υπηρεσίας Translator προσδίδει στο σύστημα ευελιξία και δυνατότητα υποστήριξης πολλαπλών γλωσσών, γεγονός που το καθιστά περισσότερο προσαρμοστικό σε διαφορετικά περιβάλλοντα και χρήσεις.

3.5.2 Βιβλιοθήκη ESP32-audioI2S-master

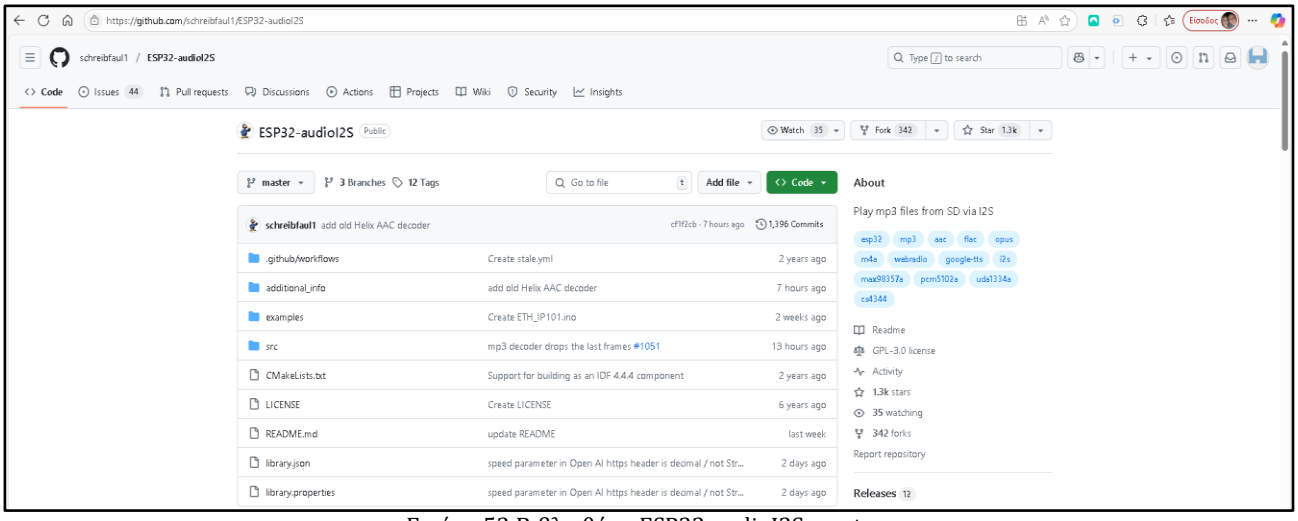
Η βιβλιοθήκη ESP32-audioI2S-master [40] αποτελεί μία εξειδικευμένη λύση για την υποστήριξη αναπαραγωγής ήχου μέσω του μικροελεγκτή ESP32, αξιοποιώντας το υποσύστημα I2S (Inter-IC Sound) για την αποδοτική και ποιοτική διαχείριση ήχου (βλ. Εικόνα 53). Η βιβλιοθήκη αυτή είναι ανοιχτού κώδικα και έχει σχεδιαστεί ώστε να είναι συμβατή με το περιβάλλον Arduino IDE, προσφέροντας εύκολη ενσωμάτωση σε εφαρμογές που απαιτούν αναπαραγωγή ήχου. Μέσω της ESP32-audioI2S παρέχεται η δυνατότητα αναπαραγωγής αρχείων ήχου διαφόρων τύπων, όπως MP3, WAV, AAC και MOD, είτε από τοπικά αποθηκευμένα αρχεία (σε SD κάρτα ή SPIFFS) είτε μέσω ροής δεδομένων από το διαδίκτυο (HTTP streaming). Η έξοδος του ήχου πραγματοποιείτε με σύνδεση εξωτερικού αποκωδικοποιητή ήχου, όπως ο MAX98357A [35]. Η βιβλιοθήκη περιλαμβάνει ένα σύνολο συναρτήσεων που διευκολύνουν:

- την αρχικοποίηση της ροής ήχου (Audio audio),
- τη ρύθμιση της έντασης (audio.setVolume(volume)),
- καθώς και την επαναλαμβανόμενη εκτέλεση της αναπαραγωγής μέσω της εντολής audio.loop(), η οποία πρέπει να καλείται διαρκώς μέσα στη συνάρτηση loop() του προγράμματος.

Επιπλέον, παρέχεται και η δυνατότητα μετατροπής κειμένου σε φωνή (Text-to-Speech) μέσω της συνάρτησης audio.connecttospeech(<Text>, <Language>). Η εντολή αυτή χρησιμοποιεί την υπηρεσία Google Translate TTS για να μετατρέψει το δοθέν κείμενο σε ηχητική μορφή. Η δεύτερη παράμετρος δηλώνει τη γλώσσα της φωνής με βάση τον κωδικό ISO (π.χ. "en" για αγγλικά, "de" για γερμανικά). Για παράδειγμα, audio.connecttospeech("Wenn die Hunde

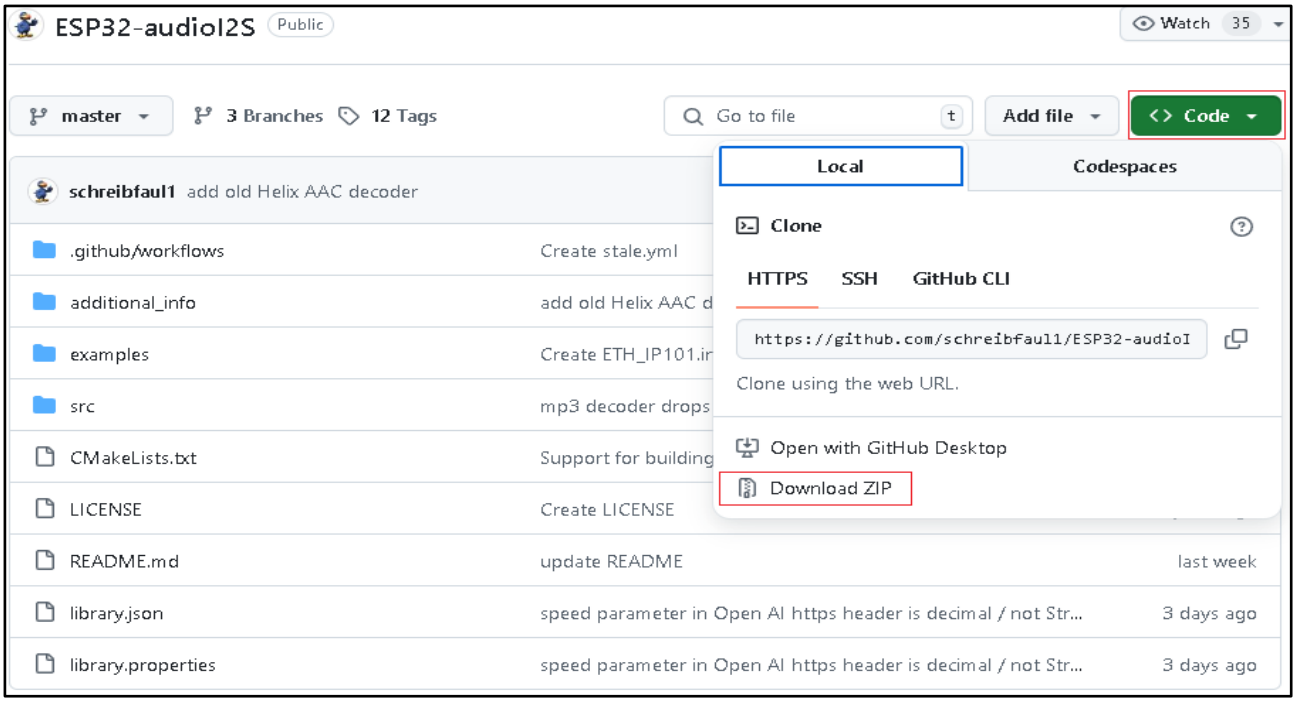
schlafen, kann der Wolf gut Schafe stehlen.", "de"); Αυτό έχει ως αποτέλεσμα την αναπαραγωγή της φράσης στα Γερμανικά μέσω του ηχείου της συσκευής.

Η χρήση της βιβλιοθήκης ESP32-audioI2S κρίθηκε απαραίτητη στο πλαίσιο της παρούσας διπλωματικής εργασίας, καθώς επιτρέπει τη μετατροπή της περιγραφής εικόνας σε ήχο, καθιστώντας εφικτή την ηχητική παρουσίαση της πληροφορίας που εξάγεται από την υπηρεσία Azure Computer Vision. Με τον τρόπο αυτό, η συσκευή που αναπτύσσεται μπορεί να παρέχει ακουστική προσβασιμότητα σε άτομα με προβλήματα όρασης, συνδυάζοντας λειτουργίες τεχνητής νοημοσύνης με τεχνολογίες φωνητικής εξόδου.



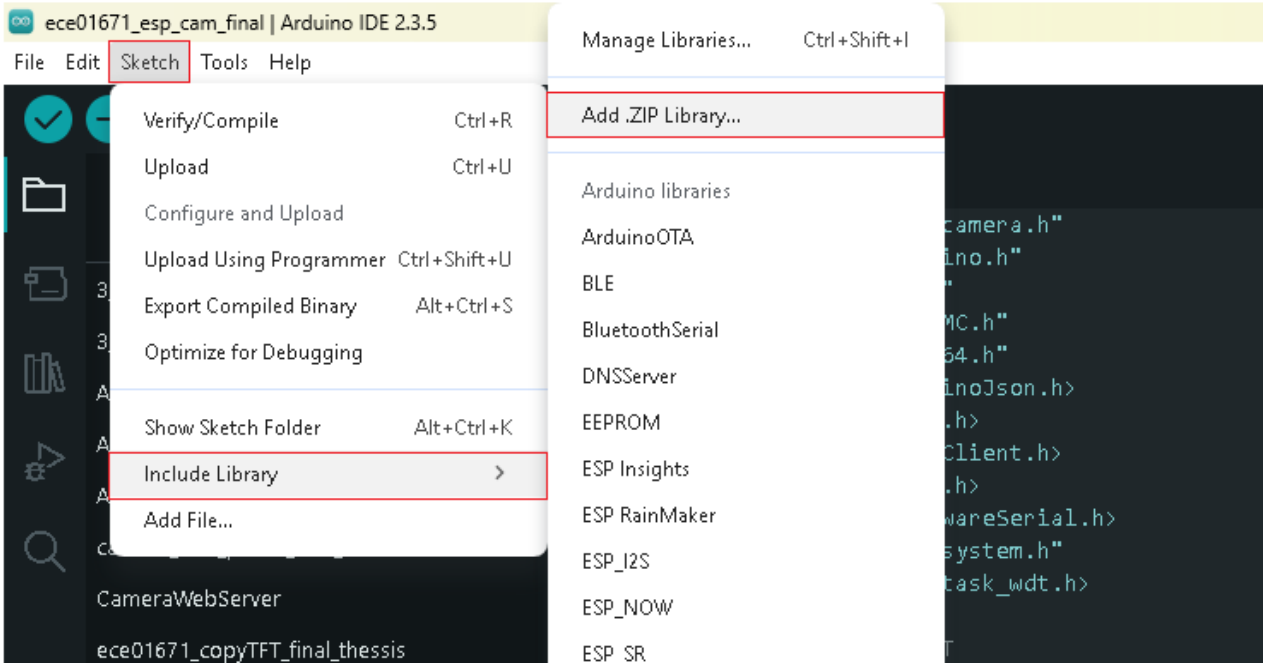
Εικόνα 53:Βιβλιοθήκη ESP32-audioI2S-master
Πηγή:https://github.com/schreibfaul1/ESP32-audioI2S

Η συγκεκριμένη βιβλιοθήκη δεν περιλαμβάνεται στις προεγκατεστημένες βιβλιοθήκες του ESP32, επομένως απαιτείται να την κατεβάσουμε χειροκίνητα από το GitHub. Η διαδικασία είναι απλή, αρχικά κάνουμε κλικ στο πράσινο κουμπί "Code", έπειτα, στο αναδυόμενο μενού που εμφανίζεται, επιλέγουμε την καρτέλα "Local" και στη συνέχεια κάνουμε κλικ στην επιλογή "Download ZIP". Με αυτόν τον τρόπο, κατεβαίνει ολόκληρο το αποθετήριο της βιβλιοθήκης σε μορφή αρχείου ZIP στον υπολογιστή (βλ. Εικόνα 54).



Εικόνα 54:Λήψη βιβλιοθήκης ESP32-audioI2S από Github

Αφού ολοκληρωθεί η λήψη του αρχείου ZIP της βιβλιοθήκης, προχωράμε στην εγκατάστασή της μέσω του περιβάλλοντος Arduino IDE. Για να το πετύχουμε αυτό, μεταβαίνουμε στο επάνω μενού Sketch, επιλέγουμε Include Library και στη συνέχεια κάνουμε κλικ στην επιλογή Add .ZIP Library. Στο παράθυρο που ανοίγει, εντοπίζουμε και επιλέγουμε το αρχείο ZIP που κατεβάσαμε από το GitHub. Η βιβλιοθήκη θα εγκατασταθεί αυτόματα και θα είναι άμεσα διαθέσιμη για χρήση (βλ. Εικόνα 55).



Εικόνα 55:Εγκατάσταση βιβλιοθήκης ESP32-audioI2S

3.6 Σύνοψη τρίτου κεφαλαίου

Στο τρίτο κεφάλαιο παρουσιάστηκαν αναλυτικά οι προδιαγραφές του συστήματος, καθώς και η διαδικασία σχεδίασης και υλοποίησης της ενσωματωμένης συσκευής που αναπτύχθηκε στο πλαίσιο της διπλωματικής εργασίας. Η συσκευή βασίστηκε στον μικροελεγκτή ESP32 και χρησιμοποιήθηκαν δύο εκδόσεις του, η ESP32-WROOM-32 και η ESP32-CAM, με δυνατότητες Wi-Fi και λήψης εικόνας/βίντεο μέσω του αισθητήρα OV2640. Για την ηχητική έξοδο χρησιμοποιήθηκε ο ενισχυτής MAX98357A, ενώ η αλληλεπίδραση με τον χρήστη επιτεύχθηκε μέσω έγχρωμης οθόνης αφής TFT 3.2'' και ποτενσιόμετρου ρύθμισης της έντασης του ήχου.

Η συσκευή ενσωμάτωσε πλήθος ηλεκτρονικών εξαρτημάτων, τα οποία διασυνδέθηκαν μέσω αξιόπιστων JST συνδέσμων και υποστηρίχθηκαν από τυπωμένο κύκλωμα (PCB). Η φυσική της δομή σχεδιάστηκε με λογισμικό 3D μοντελοποίησης και κατασκευάστηκε με εκτύπωση 3D σε υλικό PLA, διασφαλίζοντας μηχανική αντοχή, εργονομία και ευκολία συντήρησης.

Σε επίπεδο λογισμικού, αξιοποιήθηκε η πλατφόρμα Microsoft Azure και η υπηρεσία Computer Vision για την ανάλυση και ερμηνεία εικόνων μέσω τεχνητής νοημοσύνης. Επιπλέον, χρησιμοποιήθηκε η βιβλιοθήκη ESP32-audioI2S-master για την απόδοση φωνητικής εξόδου, ενισχύοντας τη λειτουργικότητα και την προσβασιμότητα της συσκευής. Η ολοκλήρωση όλων των παραπάνω στοιχείων οδήγησε σε ένα ολοκληρωμένο, διασυνδεδεμένο και λειτουργικό πρωτότυπο, το οποίο πέτυχε τον στόχο της αυτόνομης επεξεργασίας εικόνας και αναπαραγωγής πληροφορίας μέσω εικόνας και ήχου.

Κεφάλαιο 4:Επεξήγηση Λογισμικού

Ο αλγόριθμος που χρησιμοποιήθηκε στη συσκευή της διπλωματικής εργασίας αποτέλεσε τον πυρήνα της λειτουργίας της, καθώς καθόρισε τον τρόπο με τον οποίο η συσκευή επεξεργάστηκε τα δεδομένα, έλαβε αποφάσεις και εκτέλεσε συγκεκριμένες ενέργειες. Μέσα από την κατάλληλη σχεδίαση και υλοποίηση του αλγορίθμου, διασφαλίστηκε η αποδοτικότητα και η αξιοπιστία της συσκευής, ώστε να ανταποκριθεί αποτελεσματικά στον σκοπό για τον οποίο κατασκευάστηκε. Η επεξήγηση του αλγορίθμου κρίθηκε απαραίτητη για την κατανόηση της εσωτερικής λειτουργίας της συσκευής, αλλά και της συνολικής της απόδοσης.

4.1 Το λογισμικό του ESP32 CAM

Η ακόλουθη ενότητα περιλαμβάνει τρεις υποενότητες, στις οποίες παρουσιάζεται η επεξήγηση και η ανάλυση του αλγορίθμου που υλοποιείται στο ESP32 CAM. Στην πρώτη υποενότητα περιγράφονται τα αρχικά βήματα της ανάπτυξης του λογισμικού, η διαδικασία ξεκινά με την εισαγωγή των απαραίτητων βιβλιοθηκών, οι οποίες παρέχουν τη δυνατότητα διαχείρισης κρίσιμων λειτουργιών του συστήματος(π.χ. αναπαραγωγή ήχου). Στη συνέχεια, ακολουθούν οι δηλώσεις μεταβλητών και σταθερών, οι οποίες χρησιμοποιούνται για την αποθήκευση κρίσιμων δεδομένων, την υλοποίηση ελέγχου ροής (μέσω flags). Παράλληλα, δημιουργούνται αντικείμενα από τις βιβλιοθήκες (π.χ. Audio), τα οποία αξιοποιούνται αργότερα στις βασικές συναρτήσεις του προγράμματος. Τέλος, γίνεται χρήση μακροεντολών**,** που επιτρέπουν την καθορισμένη και επαναχρησιμοποιήσιμη αναφορά σε αριθμητικές τιμές, συμβάλλοντας έτσι στην καλύτερη αναγνωσιμότητα και συντήρηση του κώδικα. Στη δεύτερη υποενότητα αναλύονται οι συναρτήσεις που καλούνται μέσα στις βασικές δομές του προγράμματος, δηλαδή στις setup() και loop(). Οι συναρτήσεις αυτές αποτελούν δομικά στοιχεία του αλγορίθμου, καθώς περιλαμβάνουν εξειδικευμένες λειτουργίες που επιτρέπουν την αρχικοποίηση, τον χειρισμό περιφερειακών μονάδων και την αλληλεπίδραση με εξωτερικές υπηρεσίες. Η τρίτη και τελευταία υποενότητα επικεντρώνεται στην αναλυτική παρουσίαση των βασικών συναρτήσεων του προγράμματος: setup() και loop(), οι οποίες αποτελούν την καρδιά της λειτουργίας κάθε προγράμματος σε περιβάλλον Arduino. Η συνάρτηση setup() εκτελείται μία φορά κατά την εκκίνηση και είναι υπεύθυνη για την αρχικοποίηση του συστήματος (π.χ εκκίνηση της σειριακής επικοινωνίας). Αντίστοιχα, η loop() αποτελεί τη βασική επαναλαμβανόμενη ρουτίνα, όπου ελέγχονται συνεχώς οι είσοδοι του χρήστη (π.χ. αγγίγματα στην οθόνη), εκτελούνται εντολές ανάλογα με τη θέση αφής, πραγματοποιούνται έλεγχοι χρονισμού, ενώ παράλληλα ενεργοποιούνται οι επιμέρους λειτουργίες του συστήματος, όπως η αναπαραγωγή ραδιοφώνου, η αποστολή φωνής για αναγνώριση, ή λήψη δεδομένων από cloud υπηρεσίες.

4.2.1 Βιβλιοθήκες, δηλώσεις μεταβλητών και ορισμός μακροεντολών ESP32 CAM

Παρακάτω παρουσιάζεται η επεξήγηση και η ανάλυση του αλγόριθμου του ESP32 CAM. Στην αρχή κάθε προγράμματος ή κώδικα, η εισαγωγή βιβλιοθηκών αποτελεί ένα κρίσιμο βήμα για τη σωστή λειτουργία και ανάπτυξη της εφαρμογής. Οι βιβλιοθήκες είναι συλλογές έτοιμου κώδικα που παρέχουν προκαθορισμένες συναρτήσεις και εργαλεία, επιτρέποντας στον προγραμματιστή να αποφεύγει την επαναδημιουργία βασικών λειτουργιών. Ξεκινώντας από πρώτη επάνω βιβλιοθήκη και προχωρώντας προς τα κάτω (βλπ. Εικόνα 56), προκύπτει:

1. **esp_camera.h**: Η βιβλιοθήκη αυτή χρησιμοποιείται για τον έλεγχο της λειτουργίας της κάμερας που είναι συνδεδεμένη σε μονάδα ESP32-CAM. Παρέχει λειτουργίες για την αρχικοποίηση της κάμερας, τη λήψη φωτογραφιών, καθώς και για τη ρύθμιση παραμέτρων όπως η ανάλυση και η ποιότητα εικόνας.
2. **Arduino.h**: Βασική βιβλιοθήκη του περιβάλλοντος Arduino, η οποία περιλαμβάνει τις κύριες δηλώσεις και συναρτήσεις απαραίτητες για την ορθή εκτέλεση του προγράμματος (όπως setup(), loop(), ψηφιακή και αναλογική ανάγνωση/εγγραφή). Εξασφαλίζει τη σύνδεση του κώδικα με τον πυρήνα του συστήματος Arduino.
3. **FS.h**: Προσφέρει υποστήριξη για τη διαχείριση συστημάτων αρχείων στο ESP32. Η βιβλιοθήκη FS (File System) επιτρέπει τη δημιουργία, ανάγνωση, εγγραφή και διαγραφή αρχείων σε μέσα αποθήκευσης, όπως κάρτες SD.
4. **base64.h**: Χρησιμοποιείται για την κωδικοποίηση και αποκωδικοποίηση δεδομένων σε μορφή Base64. Η κωδικοποίηση αυτή είναι απαραίτητη για τη μετατροπή δυαδικών δεδομένων (όπως εικόνες) σε μορφή κειμένου, ώστε να μπορούν να αποσταλούν μέσω πρωτοκόλλων όπως το HTTP ή να ενσωματωθούν σε JSON αρχεία.
5. **ArduinoJson.h**: Βιβλιοθήκη για τη δημιουργία και ανάλυση (parsing) δεδομένων μορφής JSON σε περιβάλλον Arduino. Διευκολύνει την ανταλλαγή δομημένων δεδομένων με απομακρυσμένους servers, ενισχύοντας τη διαλειτουργικότητα του συστήματος.

```
1  #include "esp_camera.h"
2  #include "Arduino.h"
3  #include "FS.h"                // SD Card ESP32
4  #include "base64.h"            // Include your Base64 library
5  #include <ArduinoJson.h>       // Include ArduinoJson for JSON parsing
6  #include <WiFi.h>
7  #include <HTTPClient.h>
8  #include <time.h>
9  #include <SoftwareSerial.h>
10 #include "esp_system.h"
11 #include <esp_task_wdt.h>
```

Εικόνα 56:Εισαγωγή βιβλιοθηκών (ESP32 CAM)

6. **WiFi.h**: Παρέχει τις απαραίτητες λειτουργίες για τη σύνδεση του ESP32 σε ασύρματα δίκτυα WiFi. Υποστηρίζει τόσο τη λειτουργία Station όσο και Access Point, ενώ προσφέρει δυνατότητες διαχείρισης IP διευθύνσεων και ασφαλείας.
7. **HTTPClient.h**: Επιτρέπει στο ESP32 να λειτουργεί ως πελάτης HTTP (HTTP client), δίνοντάς του τη δυνατότητα να πραγματοποιεί αιτήματα τύπου GET, POST, PUT κ.ά. σε web servers, καθιστώντας δυνατή την επικοινωνία και αλληλεπίδραση με διαδικτυακές υπηρεσίες.
8. **time.h**: Παρέχει λειτουργίες για τη διαχείριση και συγχρονισμό ώρας και ημερομηνίας. Μέσω της βιβλιοθήκης αυτής, είναι δυνατός ο συγχρονισμός της συσκευής με εξωτερικούς NTP servers για την απόκτηση ακριβούς χρονικής πληροφορίας.
9. **SoftwareSerial.h**: Επιτρέπει τη δημιουργία επιπλέον σειριακών θυρών μέσω λογισμικού σε οποιοδήποτε διαθέσιμο ψηφιακό pin του ESP32. Αυτό είναι ιδιαίτερα χρήσιμο όταν απαιτείται παράλληλη επικοινωνία με πολλαπλές σειριακές συσκευές.

10. **esp_system.h:**Παρέχει πρόσβαση σε λειτουργίες χαμηλού επιπέδου του λειτουργικού συστήματος του ESP32, όπως έλεγχος ενέργειας, επανεκκινήσεις και άντληση πληροφοριών για το υλικό(π.χ. ESP Restart).
11. **esp_task_wdt.h**: Βιβλιοθήκη για τον έλεγχο και τη χρήση του Task Watchdog Timer (WDT) του ESP32. Ο μηχανισμός WDT ανιχνεύει καθυστερήσεις ή κολλήματα στις διεργασίες και επανεκκινεί το σύστημα για να διασφαλίσει τη συνεχή και αξιόπιστη λειτουργία του.

- **Watchdog Timer (WDT) και SoftwareSerial configuration:** Στο επόμενο κομμάτι κώδικα (βλπ. Εικόνα 57), ορίζονται κάποιες Μακροεντολές, σε μια από αυτές δηλώνεται ότι η χρονική διάρκεια timeout για το Watchdog Timer (WDT) θα είναι 5 δευτερόλεπτα και αν το πρόγραμμα κολλήσει και ενεργοποιηθεί ο WDT, θα πρέπει να γίνει υποχρεωτικό reset του ESP32. Το true σημαίνει ότι το σύστημα θα γίνει αυτόματη επανεκκίνηση. Επίσης ορίζονται οι ακροδέκτες RX και TX, συγκεκριμένα το pin GPIO14 θα χρησιμοποιηθεί ως RX (λήψη δεδομένων) και το GPIO15 ως TX (αποστολή δεδομένων) για τη σειριακή επικοινωνία μέσω λογισμικού. Τέλος, δημιουργείτε ένα αντικείμενο softSerial τύπου SoftwareSerial, το οποίο χρησιμοποιεί τα pins RX και TX που ορίσαμε (14 και 15 αντίστοιχα).

```
13  #define WDT_TIMEOUT_SECONDS 5  // Adjust timeout // Configure WDT (Timeout=5 seconds)
14  #define WDT_PANIC_RESET true   // Force reset if task hangs
15
16  // Define RX and TX pins for SoftwareSerial on ESP32-CAM
17  #define RX_PIN 14  // GPIO 14 as RX
18  #define TX_PIN 15  // GPIO 15 as TX
19  SoftwareSerial softSerial(RX_PIN, TX_PIN); // RX, TX for SoftwareSerial
```

Εικόνα 57:Ορισμός μακροεντολών (WDT,Softserial - RX/TX)

- **Δήλωση μεταβλητών SSID, PASS, Πρωτοκόλλου NTP:** Συνεχίζοντας την ανάλυση του αλγορίθμου, στην Εικόνα 58 δημιουργούνται μεταβλητές για την αποθήκευση των Wi-Fi credentials, συγκεκριμένα τις μεταβλητές ssid και password τύπου String, οι οποίες προορίζονται να αποθηκεύσουν αντίστοιχα το όνομα του δικτύου και τον κωδικό πρόσβασης. Οι μεταβλητές αυτές αρχικοποιούνται κενές ώστε να διασφαλιστεί ότι δεν περιέχουν άκυρα δεδομένα. Παράλληλα, δηλώνονται δύο λογικές μεταβλητές, usernameReceived και passwordReceived, που λειτουργούν ως σημαίες για την παρακολούθηση της λήψης των στοιχείων σύνδεσης. Αυτές αρχικά τίθενται σε false και μεταβαίνουν σε true μόλις ληφθεί σωστά το SSID και ο κωδικός πρόσβασης αντίστοιχα. Επιπλέον, για τον έλεγχο της λειτουργίας του φλας της κάμερας ESP32-CAM, χρησιμοποιούνται οι μεταβλητές flashStartTime και flashOn, όπου η πρώτη αποθηκεύει τον χρόνο ενεργοποίησης του φλας και η δεύτερη δηλώνει αν το φλας είναι ενεργό ή όχι, επιτρέποντας έτσι την ελεγχόμενη χρονικά λειτουργία του. Τέλος, ορίζονται οι παράμετροι για τον συγχρονισμό της ώρας μέσω πρωτοκόλλου NTP, με την ntpServer να καθορίζει τη διεύθυνση του διακομιστή συγχρονισμού ("pool.ntp.org"), την gmtOffset_sec να ρυθμίζει τη διαφορά ζώνης ώρας στους +7200 δευτερόλεπτα (GMT+2) και την daylightOffset_sec να λαμβάνει υπόψη τη θερινή ώρα προσθέτοντας επιπλέον 3600 δευτερόλεπτα.

```
21  String ssid = "";  // Clear previous data
22  String password = "";
23  bool usernameReceived = false;  // Flags to track data reception
24  bool passwordReceived = false;
25
26  unsigned long flashStartTime = 0;//for turn on cam flashlight
27  bool flashOn = false;
28
29  // NTP server and time zone
30  const char* ntpServer = "pool.ntp.org";
31  const long  gmtOffset_sec = 7200; // Adjust for your timezone (e.g., -3600 for GMT-1)
32  const int   daylightOffset_sec = 3600; // Adjust for daylight savings
```

Εικόνα 58:Δήλωση μεταβλητών (SSID,PASS, Πρωτοκόλλου NTP)

- **Δήλωση μεταβλητών Azure blob storage και Azure computer vision API**: Στο επόμενο κομμάτι κώδικα που απεικονίζεται στην Εικόνα 59, ορίζουμε κάποιες μεταβλητές απαραίτητες για την παραμετροποίηση του Azure Blob Storage Azure Computer Vision API. Συγκεκριμένα, σε αυτό το τμήμα του κώδικα, ορίζονται σταθερές μεταβλητές τύπου const char*, όπου αποθηκεύονται κρίσιμες πληροφορίες για τη διασύνδεση με τις υπηρεσίες του Azure Cloud.

```
// Azure Blob Storage details
const char* account_name = "ptyxiakhergasia";
const char* container_name1 = "camera-only";
const char* container_name2 = "computer-vision";
const char* blob_name = "captured_picture.jpg"; // Name of the uploaded image
String sas_token = "sv=2022-11-02&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2025-07-30T22:59:36Z&st=2025-03-03T15:59:36Z&spr=https,
// Azure Computer Vision details
const char* vision_api_key = "44f453bed3794bb49dcd1fcc318330c2";
const char* vision_endpoint = "https://pashalis.cognitiveservices.azure.com/vision/v3.2/analyze?visualFeatures=Description";
// endpoints with appropriate API version
```

Εικόνα 59:Δήλωση μεταβλητών (Azure Blob Storage και Azure Computer Vision API)

Αναλυτικότερα, η μεταβλητή account_name δηλώνει το όνομα του λογαριασμού αποθήκευσης ("ptyxiakhergasia"), ενώ οι μεταβλητές container_name1 και container_name2 καθορίζουν τα ονόματα των δύο containers που χρησιμοποιούνται: ο πρώτος προορίζεται για την απλή αποθήκευση εικόνων από την κάμερα και ο δεύτερος για τη διασύνδεση με την υπηρεσία Computer Vision. Η μεταβλητή blob_name καθορίζει την ονομασία του αρχείου εικόνας που θα αποστέλλεται και θα αποθηκεύεται στο cloud ("captured_picture.jpg"). Επιπλέον, η sas_token περιέχει το Shared Access Signature (SAS), δηλαδή ένα προκαθορισμένο κλειδί που παρέχει περιορισμένη πρόσβαση στους πόρους του λογαριασμού αποθήκευσης, με καθορισμένη ημερομηνία λήξης στις 30 Ιουλίου 2025. Παράλληλα, δηλώνονται μεταβλητές για τη σύνδεση με την υπηρεσία Azure Computer Vision. Η μεταβλητή vision_api_key αποθηκεύει το κλειδί εξουσιοδότησης για την πρόσβαση στην υπηρεσία ανάλυσης εικόνας, ενώ η vision_endpoint καθορίζει το τελικό σημείο (endpoint) του API, με συγκεκριμένες παραμέτρους που ζητούν την ανάλυση των χαρακτηριστικών περιγραφής της εικόνας (visualFeatures=Description). Οι ρυθμίσεις αυτές επιτρέπουν στο σύστημα να αποστέλλει εικόνες στην υπηρεσία Azure, να τις επεξεργάζεται μέσω τεχνητής νοημοσύνης και να ανακτά αποτελέσματα ανάλυσης σε πραγματικό χρόνο.

- **Δήλωση των ακροδεκτών (GPIO) του ESP32-CAM**: Στο παρόν τμήμα του προγράμματος (βλπ. Εικόνα 60) πραγματοποιείται αρχικά η δήλωση ενός αντικειμένου τύπου HTTPClient με το όνομα http, το οποίο χρησιμοποιείται για την πραγματοποίηση HTTP αιτημάτων προς εξωτερικούς διακομιστές κατά την εκτέλεση του αλγορίθμου. Η δυνατότητα αυτή είναι απαραίτητη για την αποστολή δεδομένων, όπως εικόνων, ή για την επικοινωνία με υπηρεσίες cloud. Στη συνέχεια, ορίζονται οι αντιστοιχίσεις των ακροδεκτών (GPIO) του ESP32-CAM με το υλικό της κάμερας, συγκεκριμένα για το μοντέλο AI-Thinker που χρησιμοποιεί αισθητήρα OV2460 ή OV5640. Τέλος, η μεταβλητή camera_fb_t *fb δηλώνεται ως δείκτης σε δομή δεδομένων που χρησιμοποιείται για την προσωρινή αποθήκευση του frame buffer, δηλαδή του στιγμιότυπου της εικόνας που συλλαμβάνεται από την κάμερα. Η διαχείριση του frame buffer είναι κρίσιμη για την περαιτέρω επεξεργασία ή αποστολή της εικόνας, καθώς επιτρέπει την αποθήκευση και ανάκτηση των δεδομένων εικόνας με αποδοτικό τρόπο.

```
45  // Global HTTP client
46  HTTPClient http;
47  // Pin definition for CAMERA_MODEL_AI_THINKER with OV5640
48  //#define PWDN_GPIO_NUM     -1 // Set to GPIO 32 if used
49  #define PWDN_GPIO_NUM      32 //with OV5640
50  #define RESET_GPIO_NUM     -1
51  #define XCLK_GPIO_NUM       0
52  #define SIOD_GPIO_NUM      26
53  #define SIOC_GPIO_NUM      27
54  #define Y9_GPIO_NUM        35
55  #define Y8_GPIO_NUM        34
56  #define Y7_GPIO_NUM        39
57  #define Y6_GPIO_NUM        36
58  #define Y5_GPIO_NUM        21
59  #define Y4_GPIO_NUM        19
60  #define Y3_GPIO_NUM        18
61  #define Y2_GPIO_NUM         5
62  #define VSYNC_GPIO_NUM     25
63  #define HREF_GPIO_NUM      23
64  #define PCLK_GPIO_NUM      22
65  camera_fb_t  *fb = NULL;
```

Εικόνα 60:Δήλωση των ακροδεκτών (GPIO) του ESP32-CAM

Πριν προχωρήσουμε στην αναλυτική παρουσίαση των βασικών συναρτήσεων void setup() και void loop(), είναι σημαντικό να προηγηθεί η μελέτη των επιμέρους συναρτήσεων που καλούνται στο εσωτερικό τους. Η κατανόηση αυτών των βοηθητικών συναρτήσεων είναι καθοριστικής σημασίας, καθώς αποτελεί θεμέλιο για την ορθή ερμηνεία της λειτουργίας και της ροής του προγράμματος.

4.1.2 Ανάλυση βοηθητικών συναρτήσεων ESP32 CAM

Προτού αναλύσουμε τις συναρτήσεις void setup() και void loop(), θα πρέπει πρώτα να αναλυθούν όλες οι άλλες συναρτήσεις που καλούνται εντός αυτών των δυο συναρτήσεων.

- Πρώτη είναι **connectToWiFi()** (βλπ. Εικόνα 61), η οποία υλοποιεί τη διαδικασία σύνδεσης της συσκευής ESP32-CAM σε ασύρματο δίκτυο Wi-Fi, λαμβάνοντας ως παραμέτρους το όνομα του δικτύου (SSID) και τον κωδικό πρόσβασης, οι παράμετροι περνιούνται ως αναφορές σε αντικείμενα τύπου String, ταυτόχρονα ελέγχεται η επιτυχής ολοκλήρωση εντός χρονικού ορίου δεκαπέντε δευτερολέπτων. Επιπλέον, κατά την προσπάθεια σύνδεσης, επανεκκινείται περιοδικά ο watchdog timer ώστε να αποτρέπεται η αυτόματη επανεκκίνηση της συσκευής λόγω καθυστέρησης. Σε περίπτωση αποτυχίας ή επιτυχίας, αποστέλλονται κατάλληλα μηνύματα τόσο στη βασική (Serial) όσο και στη δευτερεύουσα (Softserial) σειριακή θύρα, ενώ στην επιτυχή σύνδεση αποστέλλονται και η εκχωρημένη διεύθυνση IP.

```
331 void connectToWiFi(const String& ssid, const String& password){
332   const unsigned long timeout = 15000; // 15 seconds timeout
333   const unsigned long startTime = millis();
334   Serial.println("Connecting to Wi-Fi...");
335   WiFi.begin(ssid.c_str(), password.c_str()); // Convert String to const char*
336   while (WiFi.status() != WL_CONNECTED){
337     if (millis() - startTime >= timeout){
338       esp_task_wdt_reset();  // Reset watchdog
339       Serial.println("\nFailed to connect within 15 seconds!");
340       softSerial.println("WiFi connection failed (ESP32 CAM)");
341       return; // Exit the function if timeout reached
342     }
343     delay(500);
344     Serial.print(".");
345     esp_task_wdt_reset();  // Reset watchdog
346   }
347   softSerial.println("WiFi connected. (ESP32 CAM)");
348   Serial.println("\nWiFi connected. (ESP32 CAM)");
349   Serial.print("IP Address: ");
350   Serial.println(WiFi.localIP());
351 }
```

Εικόνα 61:Συνάρτηση connectToWiFi (ESP32 CAM)

- Η επόμενη συνάρτηση είναι η **describeImageWithAzure()** (βλπ. Εικόνα 62), είναι υπεύθυνη για την αποστολή αιτήματος προς την υπηρεσία Computer Vision του Microsoft Azure, με σκοπό την περιγραφή του περιεχομένου μιας εικόνας βάσει της διεύθυνσης URL της. Σε λειτουργικό επίπεδο, η συνάρτηση δέχεται ως είσοδο μία συμβολοσειρά τύπου

String, η οποία αντιστοιχεί στη προσβάσιμη διεύθυνση της εικόνας (imageUrl). Αρχικά, γίνεται έναρξη του αντικειμένου http μέσω της μεθόδου begin(), το οποίο χρησιμοποιείται για την υλοποίηση του HTTP αιτήματος. Ορίζεται ως παράμετρος το URL του Azure Vision endpoint (vision_endpoint), που είναι προκαθορισμένο σε επίπεδο μεταβλητής. Στη συνέχεια, προστίθενται δύο HTTP επικεφαλίδες (headers), η πρώτη προσδιορίζει το περιεχόμενο ως JSON (Content-Type: application/json), ενώ η δεύτερη περιέχει το μοναδικό κλειδί πρόσβασης στην υπηρεσία (Ocp-Apim-Subscription-Key), που απαιτείται για τον έλεγχο ταυτότητας. Ακολούθως, δημιουργείται το σώμα του HTTP POST αιτήματος σε μορφή JSON, το οποίο περιλαμβάνει το URL της εικόνας. Η συμβολοσειρά jsonBody περιέχει την εγγραφή {"url":"<εικόνα>"}, η οποία είναι η μορφή που αναμένει η υπηρεσία Azure. Έπειτα, η μεταβλητή httpResponseCode καταγράφει τον κωδικό απόκρισης που επιστρέφει ο διακομιστής, προσδιορίζοντας εάν η αίτηση ολοκληρώθηκε επιτυχώς (π.χ. 200 για επιτυχία). Και εφόσον η τιμή του httpResponseCode είναι θετική, δηλώνεται επιτυχής λήψη της απόκρισης και το περιεχόμενο της απάντησης αποθηκεύεται σε String μέσω της getString(). Η απάντηση αυτή αντιστοιχεί σε δομή JSON που περιέχει την περιγραφή της εικόνας, όπως προκύπτει από το AI μοντέλο της υπηρεσίας. Η απάντηση καταγράφεται και στην σειριακή θύρα για σκοπούς εντοπισμού σφαλμάτων ή παρακολούθησης. Σε περίπτωση αποτυχίας σύνδεσης ή λανθασμένου αιτήματος, ο σχετικός κωδικός σφάλματος τυπώνεται επίσης στη σειριακή θύρα. Τέλος, η σύνδεση HTTP τερματίζεται μέσω της http.end() και η συνάρτηση επιστρέφει το περιεχόμενο της απάντησης, είτε πρόκειται για τα δεδομένα JSON είτε για κενή συμβολοσειρά σε περίπτωση αποτυχίας.

```
327 String describeImageWithAzure(const String& imageUrl){
328   http.begin(vision_endpoint); // Set the Azure endpoint URL
329   http.addHeader("Content-Type", "application/json");
330   http.addHeader("Ocp-Apim-Subscription-Key", vision_api_key);
331
332   // Create the JSON payload for the request
333   String jsonBody = "{\"url\":\"" + imageUrl + "\"}";
334   int httpResponseCode = http.POST(jsonBody); // Send the request
335
336   String response = "";
337   if (httpResponseCode > 0){
338     response = http.getString(); // Get the response
339     Serial.print("Response from Azure: ");
340     Serial.println(response); // Log the response
341   }
342   else{
343     Serial.print("Error in POST request: ");
344     Serial.println(httpResponseCode); // Log the error code
345   }
346
347   http.end(); // Close the HTTP connection
348   return response; // Return the response JSON
349 }
```

Εικόνα 62:Συνάρτηση describeImageWithAzure

- Η συνάρτηση που ακολουθεί είναι η **uploadImageToBlob()** (βλπ. Εικόνα 63) και αποτελεί μία ολοκληρωμένη υλοποίηση μεταφόρτωσης δεδομένων εικόνας σε υπηρεσία αποθήκευσης Azure Blob Storage από το ESP32-CAM. Συγκεκριμένα, επιτρέπει την αποστολή δυαδικών δεδομένων (raw image bytes) σε απομακρυσμένο αποθηκευτικό χώρο στο cloud, αξιοποιώντας το REST API της Azure, υπό την παρουσία ενός έγκυρου SAS (Shared Access Signature) token. Η είσοδος της συνάρτησης περιλαμβάνει τρεις παραμέτρους: έναν δείκτη σε πίνακα δυαδικών δεδομένων εικόνας (imageData), το πραγματικό μήκος των δεδομένων (actualLength) και μία ακέραια μεταβλητή (section) η οποία καθορίζει σε ποιο container θα αποσταλεί η εικόνα, διακρίνοντας μεταξύ της λειτουργίας απλής αποθήκευσης εικόνας (camera-only) και της περαιτέρω επεξεργασίας μέσω υπολογιστικής όρασης (computer vision). Αρχικά, τα δεδομένα εικόνας κωδικοποιούνται σε μορφή Base64 μέσω της βιβλιοθήκης base64::encode, και κατόπιν

επαναποκωδικοποιούνται σε δυαδική μορφή, η οποία αποτελεί το απαιτούμενο format για το API του Azure. Η επαναποκωδικοποίηση εξυπηρετεί τεχνικές ανάγκες διαχείρισης της μνήμης σε περιβάλλον περιορισμένων πόρων, επιτρέποντας πιθανή επαλήθευση ή επεξεργασία των δεδομένων σε ενδιάμεσο στάδιο. Η URL του προορισμού δημιουργείται δυναμικά βάσει του container στόχου, ο οποίος καθορίζεται από την τιμή της section. Στην περίπτωση section == 2, προστίθεται χρονική σήμανση (time_date) στο όνομα του αρχείου ώστε να εξασφαλιστεί η μοναδικότητα και ιστορικότητα των μεταφορτώσεων. Η δομή της τελικής διεύθυνσης URL ακολουθεί το πρότυπο https://<account>.blob.core.windows.net/<container>/<filename>? <sas_token>, με όλα τα απαιτούμενα credentials ενσωματωμένα στο SAS token. Ακολούθως, ξεκινά μία σύνδεση HTTP μέσω της http.begin(blobUrl) και προστίθενται οι απαιτούμενες επικεφαλίδες: Content-Type: application/octet-stream για αποστολή δυαδικών δεδομένων και x-ms-blob-type: BlockBlob, σύμφωνα με τις προδιαγραφές της Azure για μεταφορτώσεις αρχείων blob. Η αποστολή του περιεχομένου πραγματοποιείται με τη μέθοδο PUT, η οποία αναμένεται από το Azure Blob Storage API για εγγραφή σε συγκεκριμένο blob. Το αποτέλεσμα της αίτησης ελέγχεται μέσω του httpResponseCode, με θετική τιμή να δηλώνει επιτυχή μεταφόρτωση, ενώ σε αντίθετη περίπτωση εκτυπώνεται μήνυμα σφάλματος με τον σχετικό HTTP κωδικό. Αξίζει να σημειωθεί η χρήση της συνάρτησης esp_task_wdt_reset() πριν και μετά από τις κύριες λειτουργίες της συνάρτησης, η οποία εξασφαλίζει την αποφυγή ενεργοποίησης του watchdog timer της ESP32, εφόσον η διαδικασία μεταφόρτωσης μπορεί να είναι χρονικά απαιτητική. Επιπλέον, γίνεται σωστή διαχείριση της δυναμικά εκχωρηθείσας μνήμης (delete[] decodedImageData), ώστε να αποφεύγονται σπατάλη πόρων.

```
390  String uploadImageToBlob(const uint8_t* imageData, size_t actualLength , int section) {
391    // Step 1: Encode the image data to Base64
392    String base64Image = base64::encode(imageData, actualLength);
393    esp_task_wdt_reset();  // Reset watchdog
394    // Step 2: Decode the Base64 string back to binary for Azure upload
395    size_t decodedLength = ((base64Image.length() + 3) / 4) * 3; // Calculate max size for decoded data
396    uint8_t* decodedImageData = new uint8_t[decodedLength];      // Allocate memory for the decoded image
397    size_t actualDecodedLength = base64Decode(base64Image.c_str(), decodedImageData); // Decode Base64 string to binary data
398    String blobUrl;
399    if (section==2){
400      String time_date = getDateTimeString() +"_";
401      // Construct the URL for the blob upload for CAMERA ONLY
402      blobUrl = "https://" + String(account_name) + ".blob.core.windows.net/" + container_name1 + "/" + time_date + blob_name + "?" + sas_token;
403
404    }
405    if (section==4){
406      // Construct the URL for the blob upload for COMPUTER VISION
407      blobUrl = "https://" + String(account_name) + ".blob.core.windows.net/" + container_name2 + "/" + blob_name + "?" + sas_token;
408
409    }
410    // Initialize HTTP for upload
411    http.begin(blobUrl);
412    http.addHeader("Content-Type", "application/octet-stream"); // Set content type to binary
413    http.addHeader("x-ms-blob-type", "BlockBlob"); // Required by Azure for blob upload
414    // Upload the decoded binary image data
415    int httpResponseCode = http.PUT((uint8_t*)decodedImageData, actualDecodedLength); // Use the actual decoded length
416    esp_task_wdt_reset();  // Reset watchdog
417    if (httpResponseCode > 0){
418      Serial.println("Image successfully uploaded to Blob Storage!");
419      Serial.print("Blob URL: ");
420      Serial.println(blobUrl); // Return the blob URL
421      delete[] decodedImageData; // Clean up allocated memory
422      return blobUrl; // Return the URL of the uploaded image
423    }
424    else{
425      Serial.print("Failed to upload image. HTTP Response Code: ");
426      Serial.println(httpResponseCode);
427      delete[] decodedImageData; // Clean up allocated memory
428      return "";
429    }
430
431    http.end(); // Close the HTTP connection
432  }
```

Εικόνα 63:Συνάρτηση uploadImageToBlob

- Η συνάρτηση **getDateTimeStrin**g() (βλπ. Εικόνα 64) αποτελεί μια βοηθητική ρουτίνα η οποία ανακτά την τρέχουσα τοπική ημερομηνία και ώρα από τον NTP Server (pool.ntp.org) και την επιστρέφει σε μορφοποιημένο συμβολοσειρά τύπου String. Ο κύριος σκοπός της είναι να παρέχει μοναδικές χρονικές σημάνσεις (timestamps) οι οποίες μπορούν να

χρησιμοποιηθούν για την ονομασία αρχείων ή την καταγραφή γεγονότων (logging), εξασφαλίζοντας χρονική ιχνηλασιμότητα. Η υλοποίηση βασίζεται στη χρήση της δομής tm, η οποία είναι τυπική στην ANSI C για την αναπαράσταση ημερομηνίας και ώρας. Αρχικά, μέσω της εντολής getLocalTime(&timeinfo), επιχειρείται η απόκτηση της τρέχουσας τοπικής ώρας, όπως αυτή έχει συγχρονιστεί από τον NTP server που ορίστηκε νωρίτερα στο πρόγραμμα. Εάν η προσπάθεια αποτύχει, καταγράφεται σχετικό διαγνωστικό μήνυμα στην σειριακή κονσόλα και επιστρέφεται η συμβολοσειρά "TimeUnavailable", επιτρέποντας την ασφαλή συνέχιση του προγράμματος χωρίς σφάλματα. Σε περίπτωση επιτυχούς απόκτησης της ώρας, χρησιμοποιείται η συνάρτηση strftime() για να μετατραπεί η χρονική πληροφορία σε μορφοποιημένη συμβολοσειρά σύμφωνα με το πρότυπο "%Y%m%d_%H%M%S". Το αποτέλεσμα είναι μια συμβολοσειρά όπως 20250428_173012, που δηλώνει αντίστοιχα το έτος, μήνα, ημέρα και ώρα, λεπτά, δευτερόλεπτα. Η συνάρτηση επιστρέφει την τελική συμβολοσειρά ως αντικείμενο τύπου String, το οποίο είναι ιδιαίτερα χρήσιμο σε εφαρμογές IoT που απαιτούν δυναμική δημιουργία ονομάτων αρχείων ή γεγονότων με χρονική σήμανση, όπως κατά τη μεταφόρτωση δεδομένων εικόνας

```
396 String getDateTimeString(){
397   struct tm timeinfo;
398   if (!getLocalTime(&timeinfo)){
399     Serial.println("Failed to obtain time");
400     return "TimeUnavailable";
401   }
402   // Format the date and time as a string
403   char dateTimeString[20];
404   strftime(dateTimeString, sizeof(dateTimeString), "%Y%m%d_%H%M%S", &timeinfo);
405   return String(dateTimeString);
406 }
```

Εικόνα 64:Συνάρτηση getDateTimeString

- Προτελευταία συνάρτηση είναι η **base64Decode()** (βλπ. Εικόνα 65), αποσκοπεί στην αποκωδικοποίηση δεδομένων που έχουν κωδικοποιηθεί με τον αλγόριθμο Base64, ο οποίος χρησιμοποιείται ευρέως για τη μετατροπή δυαδικών δεδομένων σε μορφή συμβολοσειράς, κατάλληλη για μεταφορά μέσω πρωτοκόλλων που υποστηρίζουν μόνο χαρακτήρες ASCII. Πιο συγκεκριμένα, η συνάρτηση λαμβάνει ως είσοδο έναν δείκτη σε συμβολοσειρά χαρακτήρων (const char* input) που περιέχει τα κωδικοποιημένα δεδομένα και έναν δείκτη σε πίνακα τύπου uint8_t (uint8_t* output) όπου θα αποθηκευτεί το αποτέλεσμα της αποκωδικοποίησης. Η διαδικασία αποκωδικοποίησης βασίζεται στη σύγκριση κάθε χαρακτήρα εισόδου με τους χαρακτήρες του πρότυπου πίνακα base64_chars, προκειμένου να προσδιοριστεί η αντίστοιχη αριθμητική του τιμή. Στη συνέχεια, μέσω διαδοχικών bitwise πράξεων, οι τιμές αυτές συνδυάζονται ώστε να ανασυντεθούν τα αρχικά bytes των δεδομένων. Ιδιαίτερη μέριμνα λαμβάνεται για την αντιμετώπιση των χαρακτήρων συμπλήρωσης ('=') που ενδέχεται να υπάρχουν στο τέλος της συμβολοσειράς, καθώς αυτοί δεν αντιστοιχούν σε πραγματικά δεδομένα και συνεπώς πρέπει να αφαιρεθούν από το συνολικό μέγεθος των αποκωδικοποιημένων δεδομένων. Η συνάρτηση επιστρέφει τον ακριβή αριθμό των byte που αποκωδικοποιήθηκαν, αφαιρώντας τα bytes που αντιστοιχούν στη συμπλήρωση.

```
size_t base64Decode(const char* input, uint8_t* output){
  const char* base64_chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
  int in_len = strlen(input);
  int i = 0, j = 0;
  int pad = 0;
  // Count padding characters
  if (input[in_len - 1] == '=') pad++;
  if (input[in_len - 2] == '=') pad++;
  for (int pos = 0; pos < in_len; pos++) {
    char c = input[pos];
    if (c == '=') break;
    // Find the index of the current character in the base64 character set
    const char* p = strchr(base64_chars, c);
    if (p) {
      int val = p - base64_chars;
      if (pos % 4 == 0) {
        output[j] = (val << 2);
      } else if (pos % 4 == 1) {
        output[j++] |= (val >> 4);
        output[j] = (val & 0x0F) << 4;
      } else if (pos % 4 == 2) {
        output[j++] |= (val >> 2);
        output[j] = (val & 0x03) << 6;
      } else if (pos % 4 == 3) {
        output[j++] |= val;
      }
    }
  }
  return j - pad; // Return the number of bytes decoded, subtracting padding
}
```

Εικόνα 65:Συνάρτηση base64Decode

- Τέλος, η συνάρτηση **receiveWiFiCredentials()** υλοποιεί τη διαδικασία λήψης και εξαγωγής των στοιχείων σύνδεσης Wi-Fi (SSID και password) μέσω σειριακής επικοινωνίας (βλπ. Εικόνα 66), χρησιμοποιώντας τη θύρα softSerial του μικροελεγκτής Αναλυτικά, η συνάρτηση εισέρχεται σε έναν επαναληπτικό βρόχο while, ο οποίος συνεχίζει να εκτελείται μέχρις ότου ληφθεί πλήρως το ζεύγος των διαπιστευτηρίων. Επίσης, καλείται σε κάθε επανάληψη η συνάρτηση esp_task_wdt_reset() για την επαναφορά του watchdog timer, ώστε να αποφεύγεται η ανεπιθύμητη επανεκκίνηση του συστήματος λόγω καθυστέρησης. Η λήψη των δεδομένων πραγματοποιείται μέσω της μεθόδου softSerial.readStringUntil('\n'), η οποία διαβάζει μία γραμμή δεδομένων έως τον χαρακτήρα αλλαγής γραμμής. Τα ληφθέντα δεδομένα αναμένονται σε μορφή SSID<TAB>Password, δηλαδή τα δύο στοιχεία χωρίζονται με τον χαρακτήρα tab (\t). Η συνάρτηση indexOf('\t') εντοπίζει το διαχωριστικό σημείο, και στη συνέχεια το ssid και το password εξάγονται με χρήση της substring() και αποθηκεύονται σε αντίστοιχες μεταβλητές τύπου String.

```
void receiveWiFiCredentials() {
  while (!passwordReceived){// Keep looping until both are received
    esp_task_wdt_reset();  // Reset watchdog
  // Check if data is available on softSerial
    if (softSerial.available()){
      String data = softSerial.readStringUntil('\n');  // Read a line of input

      // Split the data by tab character '\t'
      int tabIndex = data.indexOf('\t');  // Find the position of the first '\t'

      if (tabIndex != -1){  // If there's a tab, split the string
        // First data received: treat it as SSID (username)
        if (!usernameReceived){
          ssid = data.substring(0, tabIndex);  // Get the part before the tab
          ssid.trim();  // Trim any whitespace
          usernameReceived = true;
          Serial.println("SSID received: (" + ssid + ")");

          // Get the password (after the tab)
          password = data.substring(tabIndex + 1);  // Get the part after the tab
          password.trim();  // Trim any whitespace
          passwordReceived = true;
          Serial.println("PASSWORD received: (" + password + ")");
        }
      }
    }
  }
```

Εικόνα 66:Συνάρτηση receiveWiFiCredentials

4.1.3 Ανάλυση βασικών συναρτήσεων setup() και loop() του ESP32 CAM

Αφού πραγματοποιήθηκε η αναλυτική επεξήγηση των επιμέρους συναρτήσεων, σειρά έχει η ανάλυση του κορμού του προγράμματος που βρίσκεται εντός των συναρτήσεων **setup()** και **loop().** Οι δύο αυτές συναρτήσεις αποτελούν τον βασικό σκελετό κάθε εφαρμογής σε περιβάλλον Arduino, καθορίζοντας αφενός τις αρχικές ρυθμίσεις κατά την εκκίνηση του συστήματος (setup()), και αφετέρου τη συνεχή εκτέλεση επαναλαμβανόμενων λειτουργιών κατά τη διάρκεια λειτουργίας της συσκευής (loop()). Στη συνέχεια, θα παρουσιαστεί αναλυτικά το περιεχόμενο των δύο αυτών συναρτήσεων, με στόχο την κατανόηση της λογικής ροής του προγράμματος και της αλληλεπίδρασης μεταξύ των επιμέρους υποσυστημάτων. Ο παρακάνω κώδικας τοποθετείται εντός της συνάρτησης setup() και εκτελείται μία φορά κατά την εκκίνηση του συστήματος και πραγματοποιήσει αρχικοποίηση βασικών υποσυστημάτων της συσκευής (βλπ. Εικόνα 67), ως εξης:

- **Watchdog Timer (WDT):**Ορίζεται μια δομή esp_task_wdt_config_t η οποία καθορίζει τη συμπεριφορά του Watchdog Timer. Η μεταβλητή timeout_ms λαμβάνει την τιμή του προκαθορισμένου χρόνου αναμονής (σε χιλιοστά του δευτερολέπτου), ενώ το idle_core_mask διασφαλίζει την παρακολούθηση όλων των πυρήνων του επεξεργαστή. Η ενεργοποίηση του trigger_panic επιτρέπει την αναφορά σφάλματος σε περίπτωση αδράνειας. Η εντολή esp_task_wdt_add(NULL) προσθέτει το κύριο νήμα (main task) στο Watchdog, καθιστώντας το υπό παρακολούθηση.
- **Παραμετροποίηση WiFi:**Το σύστημα τίθεται σε λειτουργία πελάτη (WIFI_STA), που σημαίνει ότι θα επιχειρήσει να συνδεθεί σε υφιστάμενο ασύρματο δίκτυο. Επιπλέον, ρυθμίζονται προσαρμοσμένοι DNS εξυπηρετητές: ο πρώτος από το Πανεπιστήμιο Δυτικής Μακεδονίας και ο δεύτερος της Google.
- **Σειριακή Επικοινωνία:** Η κύρια σειριακή θύρα (μέσω USB, Serial) ενεργοποιείται με ταχύτητα 115200 baud για σκοπούς αποσφαλμάτωσης και επικοινωνίας με το χρήστη. Παράλληλα, γίνεται εκκίνηση της δευτερεύουσας σειριακής θύρας (softSerial) με ταχύτητα 9600 baud, επιτρέποντας τη διασύνδεση με εξωτερικές συσκευές που απαιτούν χαμηλότερες ταχύτητες, όπως αισθητήρες ή συστήματα GPS.

```
71  void setup(){
72    // Initialize Watchdog Timer with proper configuration
73    esp_task_wdt_config_t wdt_config = {
74      .timeout_ms = WDT_TIMEOUT_SECONDS * 1000,
75      .idle_core_mask = (1 << portNUM_PROCESSORS) - 1, // Watch all cores
76      .trigger_panic = true
77    };
78    esp_task_wdt_init(&wdt_config);
79    esp_task_wdt_add(NULL);  // Add main task to watchdog
80
81
82
83    WiFi.mode(WIFI_STA);
84      // Set custom DNS
85    WiFi.setDNS(
86      IPAddress(83, 212, 16, 10),  // Primary DNS (pdns1.uowm.gr)
87      IPAddress(8, 8, 8, 8)   // Secondary DNS (Google DNS)
88    );
89
90    Serial.begin(115200);        // Set up hardwareSerial at 115200 baud
91    softSerial.begin(9600);     // Set up SoftwareSerial at 9600 baud
```

Εικόνα 67:Κώδικας εντός συνάρτησης setup() 1

Σε αυτό το απόσπασμα κώδικα αρχικά καλείται η συνάρτηση esp_task_wdt_reset() (βλπ. Εικόνα 68) προκειμένου να επανεκκινήσει το watchdog timer για να αποτρέψει την αυτόματη επανεκκίνηση της συσκευής λόγω υπέρβασης χρονικού ορίου λειτουργίας. Στη συνέχεια, εκτελείται η σύνδεση σε δίκτυο Wi-Fi μέσω της συνάρτησης connectToWiFi(ssid, password), στην οποία μεταβιβάζονται τα διαπιστευτήρια που έχουν προηγουμένως ληφθεί και

αποθηκευτεί. Τέλος, χρησιμοποιείται η συνάρτηση configTime() για τον συγχρονισμό του ρολογιού του συστήματος με διακομιστή NTP, με παραμέτρους τη χρονική απόκλιση από τη ζώνη GMT (gmtOffset_sec), τη θερινή ώρα (daylightOffset_sec) και τη διεύθυνση του NTP διακομιστή (ntpServer). Η επιτυχής εκτέλεση επιβεβαιώνεται με την εμφάνιση του μηνύματος "Time synchronized", γεγονός που σηματοδοτεί ότι το σύστημα διαθέτει πλέον ακριβή χρονική πληροφορία, η οποία είναι απαραίτητη για χρονοσήμανση φωτογραφιών.

```
96    esp_task_wdt_reset();  // Reset watchdog
97    // Connect to Wi-Fi
98    connectToWiFi(ssid , password);
99    // Initialize and synchronize time with NTP server
100   configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
101   Serial.println("Time synchronized");
```

Εικόνα 68:Κώδικας εντός συνάρτησης setup() 2 - WiFi connect, Time synch

Η παρακάτω ενότητα κώδικα αφορά τη διαμόρφωση της κάμερας CAMERA_MODEL AI_THINKER, η οποία μπορεί να χρησιμοποιεί είτε τον αισθητήρα OV2640 είτε τον OV5640 (βλπ. Εικόνα 69). Συγκεκριμένα, ορίζεται και αρχικοποιείται μια μεταβλητή τύπου camera_config_t, η οποία περιλαμβάνει όλες τις απαραίτητες παραμέτρους για τη σωστή λειτουργία της κάμερας. Ορίζονται οι ακίδες (GPIO) του μικροελεγκτή που συνδέονται με τα αντίστοιχα σήματα της κάμερας: δεδομένα (pin_d0 έως pin_d7), ρολόγια (pin_xclk, pin_pclk), συγχρονισμός (pin_vsync, pin_href) και διεπαφή I2C για ρυθμίσεις (pin_sscb_sda, pin_sscb_scl). Οι ακίδες pin_pwdn και pin_reset χρησιμοποιούνται για την ενεργοποίηση ή επανεκκίνηση της κάμερας, με την τιμή -1 να υποδηλώνει ότι η αντίστοιχη λειτουργία δεν χρησιμοποιείται. Η συχνότητα του εξωτερικού ρολογιού xclk_freq_hz καθορίζεται στα 20 MHz, όπως απαιτείται για τον αισθητήρες OV2640 και OV5640. Ορίζεται επίσης η μορφή εικόνας (PIXFORMAT_JPEG) για αποδοτική αποθήκευση και μετάδοση. Στη συνέχεια, γίνεται έλεγχος για την ύπαρξη εξωτερικής μνήμης PSRAM μέσω της psramFound(). Αν υπάρχει, επιλέγεται μεγαλύτερη ανάλυση (FRAMESIZE_SXGA), καλύτερη ποιότητα JPEG (jpeg_quality = 10) και δύο frame buffers, ενώ αν δεν υπάρχει, χρησιμοποιείται μέτρια ανάλυση (FRAMESIZE_SVGA), ελαφρώς χαμηλότερη ποιότητα εικόνας και ένας buffer.

```
103   //for CAMERA_MODEL_AI_THINKER with OV2640 and OV5640
104     camera_config_t config;
105     config.ledc_channel = LEDC_CHANNEL_0;
106     config.ledc_timer = LEDC_TIMER_0;
107     config.pin_d0 = Y2_GPIO_NUM;
108     config.pin_d1 = Y3_GPIO_NUM;
109     config.pin_d2 = Y4_GPIO_NUM;
110     config.pin_d3 = Y5_GPIO_NUM;
111     config.pin_d4 = Y6_GPIO_NUM;
112     config.pin_d5 = Y7_GPIO_NUM;
113     config.pin_d6 = Y8_GPIO_NUM;
114     config.pin_d7 = Y9_GPIO_NUM;
115     config.pin_xclk = XCLK_GPIO_NUM;
116     config.pin_pclk = PCLK_GPIO_NUM;
117     config.pin_vsync = VSYNC_GPIO_NUM;
118     config.pin_href = HREF_GPIO_NUM;
119     config.pin_sscb_sda = SIOD_GPIO_NUM;
120     config.pin_sscb_scl = SIOC_GPIO_NUM;
121     config.pin_pwdn = PWDN_GPIO_NUM;  // 32 (if used)
122     config.pin_reset = RESET_GPIO_NUM;  // -1 (if not used)
123     config.xclk_freq_hz = 20000000;  // Must be 20MHz for OV5640
124     config.pixel_format = PIXFORMAT_JPEG;
125     if(psramFound()){
126       config.frame_size = FRAMESIZE_SXGA; // FRAMESIZE_ + QVGA|CIF|VGA|SVGA|XGA|SXGA|UX
127       config.jpeg_quality = 10;
128       config.fb_count = 2;
129     }
130     else{
131       config.frame_size = FRAMESIZE_SVGA;
132       config.jpeg_quality = 12;
133       config.fb_count = 1;
134     }
```

Εικόνα 69:Κώδικας εντός συνάρτησης setup() 3 - Camera configuration

Η συνάρτηση esp_camera_init(&config) καλείται με δείκτη στη δομή παραμέτρων config και επιχειρεί να ενεργοποιήσει τον αισθητήρα κάμερας (βλπ. Εικόνα 70). Επιστρέφει τιμή τύπου esp_err_t, η οποία ελέγχεται για τυχόν αποτυχία. Σε περίπτωση που η αρχικοποίηση αποτύχει (δηλαδή err != ESP_OK), εμφανίζεται μήνυμα σφάλματος στη σειριακή έξοδο, και η λειτουργία τερματίζεται με return. Αντίθετα, αν η αρχικοποίηση επιτύχει, η εκτέλεση συνεχίζεται,

εκτελείται καθυστέρηση 500ms μέσω της delay(500) (ώστε να σταθεροποιηθεί η συσκευή), και εκτυπώνεται μήνυμα επιβεβαίωσης Camera initialized.

```
136   // Init Camera
137   esp_err_t err = esp_camera_init(&config);
138   if (err != ESP_OK) {
139       Serial.printf("Camera init failed with error 0x%x", err);
140       return;
141   }
142   delay(500);
143   Serial.println("Camera initialized");
```

Εικόνα 70:Κώδικας εντός συνάρτησης setup() 4 - Camera initialization

Η παρακάτω ενότητα κώδικα αφορά τη ρύθμιση παραμέτρων λειτουργίας της κάμερας (βλπ. Εικόνα 71), μέσω του αντικειμένου τύπου sensor_t*, το οποίο αποκτάται από τη συνάρτηση esp_camera_sensor_get() (βλπ. Εικόνα 69). Ο προγραμματιστής χρησιμοποιεί μεθόδους της δομής sensor_t για να τροποποιήσει βασικές ιδιότητες της εικόνας όπως η φωτεινότητα (brightness), η αντίθεση (contrast), ο κορεσμός (saturation) και η ισορροπία λευκού (whitebal). Επιπλέον, ενεργοποιούνται ή απενεργοποιούνται αυτόματοι αλγόριθμοι ελέγχου έκθεσης (exposure_ctrl), κέρδους (gain_ctrl), αυτόματης ισορροπίας λευκού (awb_gain) και λειτουργίες επεξεργασίας εικόνας όπως μείωση θορύβου (bpc, wpc), γεωμετρική διόρθωση φακού (lenc), κάθετη/οριζόντια αντιστροφή (vflip, hmirror) και ενεργοποίηση/απενεργοποίηση του χρωματικού ρυθμιστικού (colorbar). Ο σκοπός αυτών των ρυθμίσεων είναι η βελτιστοποίηση της ποιότητας εικόνας για το εκάστοτε περιβάλλον λήψης, παρέχοντας στον χρήστη ευελιξία και ακρίβεια στην παραμετροποίηση του αισθητήρα.

```
145   sensor_t *s = esp_camera_sensor_get();
146   s->set_brightness(s, 0); // -2 to 2
147   s->set_contrast(s, 2);   // -2 to 2
148   s->set_saturation(s, -2); // -2 to 2
149   s->set_gainceiling(s, (gainceiling_t)0);  // 0 to 6
150   s->set_whitebal(s, 1);   // Enable or disable white balance
151   s->set_gain_ctrl(s, 1);  // Enable or disable gain control
152   s->set_exposure_ctrl(s, 1); // Enable or disable exposure control
153   s->set_special_effect(s, 0); // 0 to 6 (0 - No Effect, 1 - Negative, 2 - Grayscale, 3 - Red Tint, 4 - Green Tint, 5 - Blue Tint, 6 - Sepia)
154   s->set_whitebal(s, 1);       // 0 = disable , 1 = enable
155   s->set_awb_gain(s, 1);       // 0 = disable , 1 = enable
156   s->set_wb_mode(s, 0);        // 0 to 4 - if awb_gain enabled (0 - Auto, 1 - Sunny, 2 - Cloudy, 3 - Office, 4 - Home)
157   s->set_exposure_ctrl(s, 1);  // 0 = disable , 1 = enable
158   s->set_aec2(s, 0);           // 0 = disable , 1 = enable
159   s->set_ae_level(s, 0);       // -2 to 2
160   s->set_aec_value(s, 300);    // 0 to 1200
161   s->set_gain_ctrl(s, 1);      // 0 = disable , 1 = enable
162   s->set_agc_gain(s, 0);       // 0 to 30
163   s->set_bpc(s, 0);            // 0 = disable , 1 = enable
164   s->set_wpc(s, 1);            // 0 = disable , 1 = enable
165   s->set_raw_gma(s, 1);        // 0 = disable , 1 = enable
166   s->set_lenc(s, 1);           // 0 = disable , 1 = enable
167   s->set_hmirror(s, 1);        // 0 = disable , 1 = enable
168   s->set_vflip(s, 1);          // 0 = disable , 1 = enable
169   s->set_dcw(s, 1);            // 0 = disable , 1 = enable
170   s->set_colorbar(s, 0);       // 0 = disable , 1 = enable
```

Εικόνα 71:Κώδικας εντός συνάρτησης setup() 5 - Sensor parameters setup

Στο περιβάλλον του Arduino, η βασική συνάρτηση loop() εκτελείται συνεχώς και αποτελεί τον κύριο κορμό της λειτουργίας της συσκευής (βλ. Εικόνα 72). Η εκτέλεσή της ξεκινά με την κλήση της esp_task_wdt_reset(), η οποία επαναφέρει τον watchdog timer, εξασφαλίζοντας ότι η συσκευή δεν θα επανεκκινήσει αυτόματα λόγω καθυστερήσεων στην εκτέλεση του κώδικα. Αμέσως μετά, το ESP32-CAM ελέγχει αν υπάρχουν εισερχόμενα δεδομένα από το άλλο ESP32 μέσω της εντολής softSerial.available(). Εφόσον εντοπιστούν δεδομένα, αυτά διαβάζονται, μετατρέπονται από String σε ακέραιο αριθμό (int), και ανάλογα με την τιμή της μεταβλητής command (2, 3, 4 ή 5), εκτελείται η αντίστοιχη προκαθορισμένη λειτουργία.

```
173   void loop(){
174     esp_task_wdt_reset();  // Reset watchdog
175     // Check for incoming data from master (ESP32)
176     if (softSerial.available()){
177       String data_received = softSerial.readStringUntil('\n'); // Read incoming data
178       Serial.println("Received from ESP32 (Master): " + data_received);
179       // Convert String to const char* using c_str() to integer
180       int command = atoi(data_received.c_str());
181       Serial.println("int:" + String(command));
```

Εικόνα 72:Κώδικας εντός συνάρτησης loop() – WDT Reset – Softserial income data check

Η επεξήγηση των Εντολών (command) περιγράφονται παρακάτω ως εξής:

- **Εντολή 2**: Όταν η μεταβλητή command έχει την τιμή 2 (βλπ. Εικόνα 73), ενεργοποιείται η διαδικασία λήψης φωτογραφίας και αποστολής της στο Azure Blob Storage, με ταυτόχρονη ενεργοποίηση του φλας LED. Αρχικά, καλείται η esp_task_wdt_reset() για την επαναφορά του watchdog timer ώστε να αποτραπεί αυτόματη επανεκκίνηση της συσκευής λόγω καθυστερήσεων. Στη συνέχεια, μέσω της softSerial.println("2"), αποστέλλεται πίσω στη master συσκευή επιβεβαίωση λήψης της εντολής. Ακολουθεί η ενεργοποίηση του φλας με την εντολή pinMode(4, OUTPUT) και digitalWrite(4, HIGH), ενεργοποιώντας το GPIO 4. Καταγράφεται η χρονική στιγμή έναρξης με flashStartTime = millis() και τίθεται η σημαία flashOn = true για μελλοντικό χρονικό έλεγχο. Έπειτα, πραγματοποιείται λήψη φωτογραφίας με τη χρήση της esp_camera_fb_get()· σε περίπτωση αποτυχίας, η διαδικασία τερματίζεται. Εφόσον επιτύχει, η εικόνα αποστέλλεται στην υπηρεσία Azure μέσω της uploadImageToBlob (fb->buf, fb->len, command). Κατόπιν, αποστέλλεται μέσω της softSerial.print(...) μήνυμα επιτυχούς αποστολής. Ακολουθεί η αποδέσμευση του frame buffer της κάμερας με esp_camera_fb_return(fb). Τέλος, ελέγχεται αν έχουν παρέλθει 2000ms από την ενεργοποίηση του φλας, και αν ναι, αυτό απενεργοποιείται με digitalWrite(4, LOW).

```
183     if (command == 2){
184       esp_task_wdt_reset();  // Reset watchdog
185       softSerial.println("2");  // Return incoming data as a response
186       // Turn on the LED flash
187       pinMode(4, OUTPUT);
188       digitalWrite(4, HIGH);
189       flashStartTime = millis();  // Store the start time
190       flashOn = true;
191       // Capture the image from the camera
192       camera_fb_t* fb = esp_camera_fb_get();
193       if (!fb){
194           Serial.println("Camera capture failed");
195           digitalWrite(4, LOW);  // Ensure LED is turned off in case of failure
196           flashOn = false;
197           return;
198       }
199       // Upload the image
200       String imageUrl = uploadImageToBlob(fb->buf, fb->len, command);
201       Serial.print("The url_2: " + imageUrl);
202       softSerial.print("7~THE PHOTO UPLOADED TO AZURE CONTAINER:(CAMERA-ONLY) SUCCESSFULLY`");
203       esp_camera_fb_return(fb);  // Release the frame buffer
204       // Check if 2 seconds have passed since turning on the flash
205       if (flashOn && millis() - flashStartTime >= 2000){
206         digitalWrite(4, LOW);  // Turn off the flash LED
207         flashOn = false;
208       }
209     }
```

Εικόνα 73:Κώδικας εντός συνάρτησης loop() – Command 2

- **Εντολή 3:** Όταν η μεταβλητή command ισούται με 3 (βλπ. Εικόνα 74), εκτελείται μια προγραμματισμένη αλλά ακόμη μη υλοποιημένη λειτουργία, η οποία προορίζεται για μελλοντική αναβάθμιση του συστήματος. Αρχικά, καλείται η esp_task_wdt_reset() για την επαναφορά του watchdog timer και την αποφυγή αυτόματης επανεκκίνησης λόγω καθυστέρησης. Στη συνέχεια, μέσω της softSerial.println("3"), αποστέλλεται επιβεβαίωση λήψης της εντολής πίσω στη master συσκευή. Ταυτόχρονα, τόσο στην κύρια θύρα Serial, όσο και μέσω της softSerial, αποστέλλεται μήνυμα που ενημερώνει ότι η συγκεκριμένη λειτουργία θα υλοποιηθεί σε μελλοντική έκδοση του έργου.

```
211     if (command==3){
212       esp_task_wdt_reset();  // Reset watchdog
213       softSerial.println("3");//Return incoming data as a response
214       Serial.println("THIS FUNCTION WILL BE A FUTURE UPDATE OF THE PROJECT");
215       softSerial.println("THIS FUNCTION WILL BE A FUTURE UPDATE OF THE PROJECT");
216     }
```

Εικόνα 74:Κώδικας εντός συνάρτησης loop() – Command 3

- **Εντολή 4:** Όταν η τιμή της μεταβλητής command ισούται με 4 (βλπ. Εικόνα 75), το σύστημα εκτελεί μια σύνθετη διαδικασία λήψης φωτογραφίας, αποστολής της στο Azure Blob Storage και περιγραφής του περιεχομένου της εικόνας μέσω της υπηρεσίας Azure Computer Vision. Αρχικά, η συνάρτηση esp_task_wdt_reset() εξασφαλίζει την επαναφορά

του watchdog timer ώστε να αποτραπεί επανεκκίνηση της συσκευής λόγω καθυστέρησης. Στη συνέχεια, αποστέλλεται μήνυμα επιβεβαίωσης της εντολής μέσω softSerial.println("4"). Ακολουθεί ενεργοποίηση του φλας LED με χρήση του GPIO 4 και καταγραφή του χρόνου ενεργοποίησής του μέσω millis(). Η φωτογραφία λαμβάνεται με τη συνάρτηση esp_camera_fb_get(), ενώ σε περίπτωση αποτυχίας η λειτουργία τερματίζεται και το φλας απενεργοποιείται. Εφόσον η λήψη επιτύχει, η εικόνα αποστέλλεται στο Azure Blob Storage μέσω της uploadImageToBlob(), και στη συνέχεια απελευθερώνεται η μνήμη του buffer με esp_camera_fb_return(). Εάν η αποστολή είναι επιτυχής, το σύστημα προχωρά στην ανάλυση της εικόνας με την describeImageWithAzure(), η οποία επιστρέφει μια JSON περιγραφή. Το JSON αποκωδικοποιείται με χρήση της deserializeJson() και εξάγονται χρήσιμα μεταδεδομένα, όπως το περιγραφικό κείμενο της εικόνας, το ποσοστό εμπιστοσύνης (confidence), και οι διαστάσεις (ύψος και πλάτος). Τέλος, όλες οι πληροφορίες προωθούνται πίσω στη master συσκευή σε μορφή συμβολοσειράς μέσω softSerial, παρέχοντας μια πλήρη περιγραφή του τι απεικονίζεται στη φωτογραφία.

```
if (command==4){
  esp_task_wdt_reset();  // Reset watchdog
  softSerial.println("4");//Return incoming data as a response
  //turn on the LED flash
  pinMode(4, OUTPUT);
  digitalWrite(4, HIGH);  // Turn on the flash LED
  flashStartTime = millis();  // Store the start time
  flashOn = true;
  // Capture the image from the camera
  camera_fb_t* fb = esp_camera_fb_get();
  if (!fb){
    Serial.println("Camera capture failed");
    digitalWrite(4, LOW);  // Ensure LED is turned off in case of failure
    flashOn = false;
    return;
  }
  // Check if 2 seconds have passed since turning on the flash
  if (flashOn && millis() - flashStartTime >= 2000){
    digitalWrite(4, LOW);  // Turn off the flash LED
    flashOn = false;
  }
  String imageUrl  = uploadImageToBlob(fb->buf, fb->len , command); // Use raw image data);
  Serial.println("The url_4: " + imageUrl);
  esp_camera_fb_return(fb); // Release the frame buffer
  if (imageUrl != ""){
    // Describe the image using Azure Computer Vision
    String description = describeImageWithAzure(imageUrl);
    Serial.println("Final Image Description: ");
    Serial.println(description); // Show the image description
    // Convert description String to const char*
    const char* json = description.c_str();
    // Allocate a temporary JsonDocument
    StaticJsonDocument<1024> doc;
    // Deserialize the JSON document
    DeserializationError error = deserializeJson(doc, json);
    if (error){
      Serial.print(F("deserializeJson() failed: "));
      Serial.println(error.f_str());
      return;
    }

    // Extracting the text and confidence
    const char* text = doc["captionResult"]["text"];
    float confidence = doc["captionResult"]["confidence"];

    // Extracting height and width
    int height = doc["metadata"]["height"];
    int width = doc["metadata"]["width"];
    // Printing the results
    Serial.println("Text: " + String(text));
    Serial.println("Confidence: " + String(confidence, 2)); // Use 2 decimal places for clarity
    Serial.println("Height: " + String(height));
    Serial.println("Width: " + String(width));
    softSerial.println(String(6) + "/" + String(confidence, 2) + "@" + String(height) + "$" + String(width) + "?" + String(text) + "%");
  }
  else{
    Serial.println("Image upload failed, skipping description.");
  }
}
```

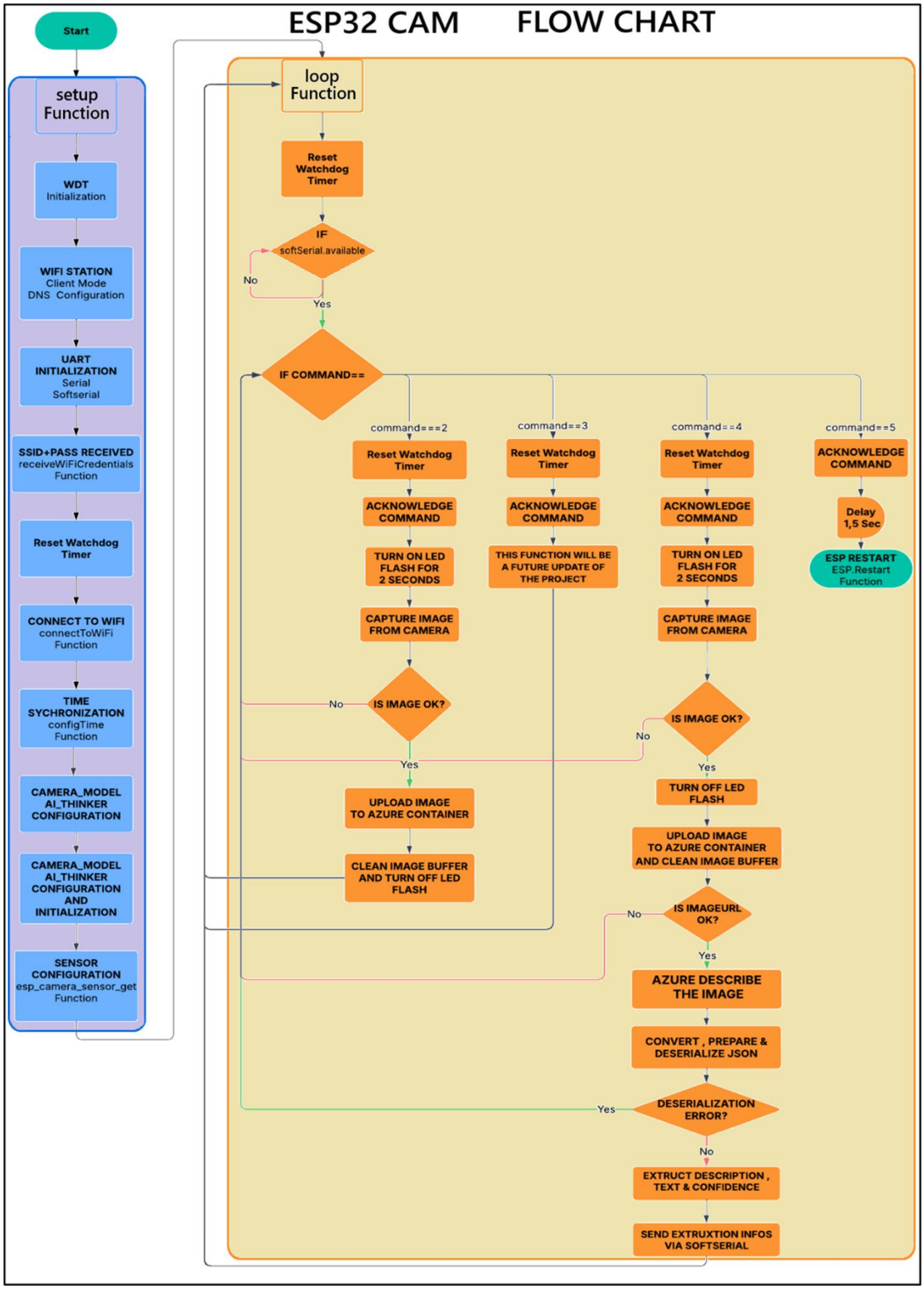
Εικόνα 75:Κώδικας εντός συνάρτησης loop() – Command 4

- **Εντολή 5:** Όταν η εντολή που λαμβάνει η συσκευή (command) είναι ίση με 5, ενεργοποιείται η διαδικασία επανεκκίνησης του ESP32-CAM. Αρχικά, η συσκευή στέλνει πίσω μέσω της σειριακής επικοινωνίας (softSerial.println("5")) έναν κωδικό επιβεβαίωσης λήψης της εντολής. Στη συνέχεια, εμφανίζεται μήνυμα στη σειριακή κονσόλα (Serial.println("ESP32 CAM RESTART")) για ενημέρωση του χρήστη ότι ξεκινά επανεκκίνηση. Ακολουθεί καθυστέρηση 1,5 δευτερολέπτου (delay(1500)) για να εξασφαλιστεί ότι όλες οι εκκρεμείς εντολές και διαδικασίες έχουν ολοκληρωθεί, και τέλος εκτελείται η εντολή ESP.restart() για την επανεκκίνηση της συσκευής (βλπ. Εικόνα 76).

```
    if (command==5){
      softSerial.println("5");
      Serial.println("ESP32 CAM RESTART");
      delay(1500);
      ESP.restart();
    }
  digitalWrite(4, LOW);  // Turn off the flash LED
  }
}
```

Εικόνα 76:Κώδικας εντός συνάρτησης loop() – Command 5

Για καλύτερη κατανόηση της λειτουργίας και των δυο συναρτήσεων setup() και loop() ακολουθεί ένα διάγραμμα ροής, το οποίο είναι ένα γραφικό εργαλείο που χρησιμοποιείται για να απεικονίσει του αλγορίθμου, βήμα προς βήμα (βλπ. Σχήμα 19).



Σχήμα 19:Διάγραμμα ροής συναρτήσεων setup() και loop() (ESP32 CAM)

Ολοκληρώνοντας την ανάλυση του αλγορίθμου λειτουργίας του ESP32-CAM, διαπιστώνεται ότι η συσκευή εκτελεί μια σειρά από έξυπνες ενέργειες, ανταποκρινόμενη σε εντολές που λαμβάνει από άλλο ESP32. Μέσα από τον κατάλληλο έλεγχο εντολών, κάνει διαχείριση του φλας, λήψη και αποστολή φωτογραφιών, καθώς και ενσωμάτωση υπηρεσιών cloud όπως το Azure Computer Vision.

4.2 Το λογισμικό του ESP32

Η παρούσα ενότητα περιλαμβάνει τρεις υποενότητες και επικεντρώνεται στην επεξήγηση και ανάλυση του αλγορίθμου που υλοποιείται στη δεύτερη μονάδα ESP32. Στην πρώτη υποενότητα περιγράφονται τα αρχικά στάδια του λογισμικού. Η ανάπτυξη ξεκινά με την εισαγωγή των αναγκαίων βιβλιοθηκών, οι οποίες επιτρέπουν την υλοποίηση βασικών λειτουργιών του συστήματος, όπως για παράδειγμα η επεξεργασία ήχου ή η διαχείριση της οθόνης. Ακολουθούν οι δηλώσεις μεταβλητών και σταθερών, που αξιοποιούνται για την αποθήκευση δεδομένων και τον έλεγχο της ροής εκτέλεσης μέσω κατάλληλων flags. Επιπλέον, δημιουργούνται αντικείμενα που προέρχονται από τις ενσωματωμένες βιβλιοθήκες (π.χ. audio) τα οποία διαδραματίζουν κρίσιμο ρόλο στην εκτέλεση των κύριων λειτουργιών του προγράμματος. Τέλος, γίνεται χρήση μακροεντολών για τη σαφή αναφορά σε σταθερές αριθμητικές τιμές, γεγονός που διευκολύνει τόσο την κατανόηση όσο και τη συντήρηση του κώδικα. Η δεύτερη υποενότητα εστιάζει στην ανάλυση των επιμέρους συναρτήσεων που χρησιμοποιούνται εντός των βασικών δομών του προγράμματος, δηλαδή των setup() και loop(). Οι βοηθητικές αυτές συναρτήσεις επιτελούν εξειδικευμένες λειτουργίες, όπως την αρχικοποίηση περιφερειακών συσκευών, την επεξεργασία δεδομένων εισόδου ή την επικοινωνία με εξωτερικές πλατφόρμες. Στην τρίτη και τελευταία υποενότητα γίνεται λεπτομερής ανάλυση των δύο βασικών συναρτήσεων του προγράμματος: setup() και loop(), οι οποίες καθορίζουν τη δομή και τον συνεχή κύκλο εκτέλεσης σε ένα περιβάλλον ανάπτυξης τύπου Arduino. Η setup() εκτελείται μία φορά κατά την εκκίνηση του συστήματος και περιλαμβάνει ενέργειες όπως η εκκίνηση της σειριακής επικοινωνίας, η σύνδεση στο δίκτυο Wi-Fi και η αρχικοποίηση των απαραίτητων μονάδων. Η loop(), από την άλλη πλευρά, είναι υπεύθυνη για την επαναλαμβανόμενη εκτέλεση εντολών, την παρακολούθηση της αλληλεπίδρασης του χρήστη με την οθόνη αφής, την υλοποίηση χρονισμένων ενεργειών, καθώς και την ενεργοποίηση επιμέρους λειτουργιών όπως το διαδικτυακό ραδιόφωνο, η φωνητική αναγνώριση ή η αλληλεπίδραση με cloud υπηρεσίες.

4.2.1 Βιβλιοθήκες, δηλώσεις μεταβλητών και ορισμός μακροεντολών ESP32

Ο παρακάτω κώδικας περιλαμβάνει τις απαραίτητες βιβλιοθήκες για τη σωστή λειτουργία του ESP32, διασφαλίζοντας τη συνδεσιμότητα, τον χειρισμό αρχείων, την επεξεργασία ήχου, την οθόνη αφής, και την προστασία συστήματος μέσω watchdog timer. Ακολουθεί μια σύντομη επεξήγηση των βιβλιοθηκών (βλπ. Εικόνα 77):

1. **mbedtls/base64.h**: Παρέχει λειτουργίες για αποκωδικοποίηση Base64, χρήσιμες για μεταφορά δεδομένων όπως εικόνες ή ήχος.
2. **Audio.h**: Υποστήριζει την επεξεργασία και αναπαραγωγή ήχου, καθώς και την μετατροπή κειμένου σε ομιλία μέσω Google Text-to-Speech (TTS).
3. **WiFi.h & WiFiClientSecure.h**: Κάνει διαχείριση ασύρματης σύνδεσης και ασφαλούς επικοινωνίας μέσω HTTP και HTTPS.
4. **Arduino.h**: Περιλαμβάνει βασικές συναρτήσεις και τύπους για το περιβάλλον Arduino.
5. **EEPROM.h**: Διαχειρίζεται την μη πτητικής μνήμης EEPROM για αποθήκευση δεδομένων.
6. **FS.h**: Παρέχει πρόσβαση στο SPIFFS (SPI Flash File System) για αποθήκευση δεδομένων σε flash.

7. **SPI.h**: Πρωτόκολλο επικοινωνίας SPI, χρησιμοποιείται για συσκευές όπως οθόνες ή αισθητήρες Touch.
8. **SoftwareSeria**l.h: Δημιουργεί εικονική σειριακή επικοινωνία (π.χ. για επικοινωνία μεταξύ δύο ESP32).
9. **TFT_eSPI.h**: Βιβλιοθήκη για χειρισμό και απεικόνιση TFT οθονών με χρήση SPI.
10. **XPT2046_Touchscreen.h**: Βιβλιοθήκη, που υποστηρίζει οθόνες αφής, που βασίζονται στο chip XPT2046. Το chip δέχεται το σήμα από την επιφάνεια αφής και επιστρέφει στο ESP32 τιμές X και Y.
11. **esp_system.h**: Περιλαμβάνει λειτουργίες χαμηλού επιπέδου για χειρισμό του συστήματος ESP32(π.χ. ESP Restart).
12. **esp_task_wdt.h**: Υποστήριξη για τον watchdog timer, ώστε να αποτρέπεται το "κόλλημα" του συστήματος.

```
#include "mbedtls/base64.h"  // Include the ESP32 built-in Base64 decoder
#include <Audio.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "Arduino.h"
#include <EEPROM.h>
#include "FS.h"   // Calibration data is stored in SPIFFS so we need to include it
#include <SPI.h>
#include <SoftwareSerial.h>
#include <TFT_eSPI.h> // Hardware-specific library
#include "Audio.h"
#include <XPT2046_Touchscreen.h>
#include "esp_system.h"
#include <esp_task_wdt.h>
```

Εικόνα 77:Εισαγωγή βιβλιοθηκών (ESP32)

Το παρακάτω απόσπασμα κώδικα έχει κύριο στόχο τη μέτρηση τάσης, τη χρήση οθόνης αφής και τη διαχείριση σειριακής επικοινωνίας. Αναλυτικότερα, ορίζονται κάποιες Μακροεντολές και μεταβλητές ως εξής:

- **Watchdog Timer (WDT):** δηλώνεται ότι η χρονική διάρκεια timeout για το Watchdog Timer (WDT) θα είναι 5 δευτερόλεπτα και αν το πρόγραμμα κολλήσει και ενεργοποιηθεί ο WDT, θα πρέπει να γίνει υποχρεωτικό reset του ESP32. Το true σημαίνει ότι το σύστημα θα γίνει αυτόματη επανεκκίνηση (βλπ. Εικόνα 78).

```
// Configure WDT (Timeout: 5 seconds)
#define WDT_TIMEOUT_SECONDS 5  // Adjust timeout as needed
#define WDT_PANIC_RESET true   // Force reset if task hangs
```

Εικόνα 78:Ορισμός μακροεντολής (WDT)

- **Μέτρηση Τάσης:** Γίνεται η μέτρηση της τάσης, χρησιμοποιούνται τα GPIO27 ως pin ελέγχου και GPIO34 ως είσοδος ADC, με χρονικό διάστημα μέτρησης κάθε 1,5 λεπτό (90.000ms) (βλπ. Εικόνα 79).

```
// Pin definitions for voltage measurement
const int CONTROL_PIN = 27;     // GPIO27 - control pin
const int VOLTAGE_PIN = 34;     // GPIO34 - ADC input
// Time interval (in milliseconds)
const unsigned long MEASURE_INTERVAL = 90000;  // 1,5 minutes
```

Εικόνα 79:Δήλωση μεταβλητής (Voltage measurement)

- **EEPROM - WiFi Credentials:** Αποθηκεύεται το SSID και ο κωδικός Wi-Fi, δημιουργείτε στην EEPROM χώρος μεγέθους 64 bytes, με καθορισμένες διευθύνσεις για το καθένα (0 για SSID και 32 για τον κωδικό) (βλπ. Εικόνα 80).

```
// Define EEPROM size and addresses
#define EEPROM_SIZE 64 // Adjust based on your requirements
#define SSID_ADDR 0    // Starting address for SSID
#define PASS_ADDR 32   // Starting address for password
```

Εικόνα 80:Ορισμός μακροεντολής (EEPROM)

- **Ρύθμιση Debounce:** έχει προβλεφθεί καθυστέρηση debounce 350ms για την αποφυγή ψευδών ενεργοποιήσεων από κουμπιά αφής (βλπ. Εικόνα 81).

```
#define DEBOUNCE_DELAY 350 // debounce delay in milliseconds
unsigned long lastMeasureTime = 0; // stores the last time a button pressed
```

Εικόνα 81:Ορισμός μακροεντολής (Debounce)

- **SoftwareSerial (UART2):** ορίζονται οι ακροδέκτες RX και TX, συγκεκριμένα το pin GPIO14 θα χρησιμοποιηθεί ως RX (λήψη δεδομένων) και το GPIO15 ως TX (αποστολή δεδομένων) για τη σειριακή επικοινωνία μέσω λογισμικού. Τέλος, δημιουργείτε ένα αντικείμενο softSerial τύπου SoftwareSerial, το οποίο χρησιμοποιεί τα pins RX και TX που ορίσαμε (14 και 15 αντίστοιχα). (βλπ. Εικόνα 82).

```
// Define RX and TX pins for SoftwareSerial on ESP32-CAM
#define RX_PIN 16  // GPIO 16 as RX(UART2 RX)
#define TX_PIN 17  // GPIO 17 as TX(UART2 TX)
SoftwareSerial softSerial(RX_PIN, TX_PIN); // RX, TX for SoftwareSerial
```

Εικόνα 82:Ορισμός μακροεντολής (SoftwareSerial - RX/TX)

- **Οθόνη Αφής TFT με XPT2046:** δημιουργεί ένα αντικείμενο με το όνομα tft βασισμένο στην κλάση TFT_eSPI, που παρέχεται από τη βιβλιοθήκη TFT_eSPI**,** για την διαχείριση της Οθόνη Αφής και τον έλεγχο αφής και με αποθήκευση των δεδομένων βαθμονόμησης αφής στο αρχείο /TouchCalData3. Η μεταβλητή REPEAT_CAL ελέγχει αν θα επαναληφθεί η βαθμονόμηση της αφής, η οποία εδώ έχει ρυθμιστεί να μην επαναλαμβάνεται (βλπ. Εικόνα 83).

```
// Touch handling for XPT2046 based screens is handled by
// the TFT_eSPI library.
TFT_eSPI tft = TFT_eSPI();       // Invoke custom library
// This is the file name used to store the touch coordinate
// calibration data. Change the name to start a new calibration.
#define CALIBRATION_FILE "/TouchCalData3"
// Set REPEAT_CAL to true instead of false to run calibration
// again, otherwise it will only be done once.
// Repeat calibration if you change the screen rotation.
#define REPEAT_CAL false
```

Εικόνα 83:Ορισμός μακροεντολής (TFT_eSPI, TFT Calibration)

- **Radio URLs:** Το παρακάτω τμήμα κώδικα περιέχει μια συλλογή από URLs που αντιστοιχούν σε ζωντανές ροές (streams) ραδιοφωνικών σταθμών μέσω διαδικτύου, κυρίως από την περιοχή της Κοζάνης αλλά και μερικούς πανελλαδικής εμβέλειας σταθμούς. Κάθε μεταβλητή τύπου const char* αποθηκεύει το URL μιας ραδιοφωνικής μετάδοσης, το οποίο μπορεί να χρησιμοποιηθεί από τον ESP32 για την αναπαραγωγή του ήχου μέσω διαδικτύου. Οι διευθύνσεις αυτές λειτουργούν ως σημεία πρόσβασης σε live audio streams σε μορφή MP3 (βλπ. Εικόνα 84).

```
// URLS of the internet radio stream
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL01 = "http://176.31.120.166:6404/stream"; //DIVA 91.6 FM KOZANI
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL02 = "http://netradio.live24.gr/realfm";//REAL 97.8 FM KOZANI
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL03 = "http://i4.streams.ovh:8078/stream?type=http&nocache=422";//NRG 89.5 FM KOZANI
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL04 = "http://netradio.live24.gr/laikos876";//LAIKOS 87.6 FM KOZANI
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL05 = "http://radiostreaming.ert.gr/ert-kozani";//ERT 100.2 FM KOZANI
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL06 = "http://178.33.135.244:3062/stream";//FRESH 92.9 FM KOZANI
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL07 = "http://netradio.live24.gr/skai1003"; //Skai 100.3 FM
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL08 = "http://stream.radiojar.com/aw1w2xfx1vduv";//Rythmos 94.9 FM
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL09 = "http://netradio.live24.gr/loveradio-1000";//Love Radio 97.5 FM
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL10 = "http://pepper966.live24.gr/pepper9660";// Pepper 96.6
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL11 = "http://metropolis.live24.gr/metropolis955thess";// Metropolis 95.5
//-------------------------------------------------------------------------------------------------------------------//
const char* streamURL12 = "http://n0e.radiojar.com/083wqknmsuhvv?rj-ttl=5&rj-tok=AAABhHgFMdoAMjpYSw5BtqF9Zg";//Rainbow 89
//-------------------------------------------------------------------------------------------------------------------//
```

Εικόνα 84:Δήλωση μεταβλητών (Radio URLs)

- **Ρυθμίσεις εξόδου ήχου με I2S**: Το I2S είναι ένα ψηφιακό πρωτόκολλο που χρησιμοποιείται για τη μετάδοση ήχου υψηλής ποιότητας. Ορίζονται οι αντίστοιχες γραμμές GPIO του ESP32 για τη μεταφορά του ήχου προς έναν εξωτερικό DAC (βλπ. Εικόνα 85).

```
// Configure I2S output for audio
#define I2S_DOUT      22
#define I2S_BCLK      26
#define I2S_LRC       25
```

Εικόνα 85:Ορισμός μακροεντολής (I2S)

- **Έλεγχος έντασης με ποτενσιόμετρο:** Το GPIO 39 είναι αναλογική είσοδος (ADC) και διαβάζεται η τιμή του ποτενσιόμετρου, η οποία χρησιμοποιείται για να καθοριστεί η ένταση του ήχου. Η μεταβλητή volume χρησιμοποιείται για να αποθηκεύει την τιμή που διαβάζεται, ώστε να χρησιμοποιηθεί μετέπειτα από analogRead(volControl) και χαρτογράφηση με map()(βλπ. Εικόνα 86).

```
// Define volume control pot connection
// ADC3 is GPIO 39
const int volControl = 39;
int volume = 6; // Integer for volume level

```

Εικόνα 86:Δήλωση μεταβλητών (Volume control)

- **FRAME_X / Y / W / H:** Ορίζονται οι σταθερές FRAME_Xn, FRAME_Yn, FRAME_Wn, FRAME_Hn, που δηλώνουν τη θέση (X, Y) και το μέγεθος (πλάτος W, ύψος H) σχημάτων επι της οθόνης (βλπ. Εικόνα 87).

```
// Switch position and size INTERNET RADIO
#define FRAME_X1 0
#define FRAME_Y1 0
#define FRAME_W1 140
#define FRAME_H1 60
// Switch position and size CAMERA ONLY
#define FRAME_X2 0
#define FRAME_Y2 65
#define FRAME_W2 140
#define FRAME_H2 60
// Switch position and size AZURE VOICE RECOGNITION
#define FRAME_X3 0
#define FRAME_Y3 130
#define FRAME_W3 140
#define FRAME_H3 60
// Switch position and size AZURE COMPUTER VISION
#define FRAME_X4 0
#define FRAME_Y4 195
#define FRAME_W4 140
#define FRAME_H4 60
```

Εικόνα 87:Ορισμός μακροεντολής (FRAME_X / Y / W / H)

- **RED-GREENBUTTON_X / Y / W / H:** Ορίζονται οι σταθερές RED-GREENBUTTON _Xn, RED-GREENBUTTON _Yn, RED-GREENBUTTON _Wn, RED-GREENBUTTON _Hn, που δηλώνουν τη θέση (X, Y) και το μέγεθος (πλάτος W, ύψος H) διακρατικών κουμπιών επι της οθόνης (βλπ. Εικόνα 88).

```
112  //Red zone size1-------------------  132  //Red zone size3-------------------
113  #define REDBUTTON_X1 141              133  #define REDBUTTON_X3 141
114  #define REDBUTTON_Y1 0                134  #define REDBUTTON_Y3 130
115  #define REDBUTTON_W1 50               135  #define REDBUTTON_W3 50
116  #define REDBUTTON_H1 60               136  #define REDBUTTON_H3 60
117  //Green zone size1                    137  //Green zone size3
118  #define GREENBUTTON_X1 191            138  #define GREENBUTTON_X3 191
119  #define GREENBUTTON_Y1 0              139  #define GREENBUTTON_Y3 130
120  #define GREENBUTTON_W1 50             140  #define GREENBUTTON_W3 50
121  #define GREENBUTTON_H1 60             141  #define GREENBUTTON_H3 60
122  //Red zone size2-------------------  142  //Red zone size4-------------------
123  #define REDBUTTON_X2 141              143  #define REDBUTTON_X4 141
124  #define REDBUTTON_Y2 65               144  #define REDBUTTON_Y4 195
125  #define REDBUTTON_W2 50               145  #define REDBUTTON_W4 50
126  #define REDBUTTON_H2 60               146  #define REDBUTTON_H4 60
127  //Green zone size2                    147  //Green zone size2
128  #define GREENBUTTON_X2 191            148  #define GREENBUTTON_X4 191
129  #define GREENBUTTON_Y2 65             149  #define GREENBUTTON_Y4 195
130  #define GREENBUTTON_W2 50             150  #define GREENBUTTON_W4 50
131  #define GREENBUTTON_H2 60             151  #define GREENBUTTON_H4 60
```

Εικόνα 88:Ορισμός μακροεντολής (RED-GREENBUTTON_X / Y / W / H)

- **FLAGS, AUDIO AND WiFiClientSecure:** Στο πλαίσιο διαχείρισης της ροής λειτουργιών του συστήματος, χρησιμοποιούνται μεταβλητές τύπου int που λειτουργούν ως **σημαίες ελέγχου (flags)**. Παράλληλα, για την αναπαραγωγή ήχου μέσω διαδικτύου (π.χ. ραδιοφωνική ροή ή υπηρεσίες TTS), αξιοποιείται το αντικείμενο Audio, το οποίο προέρχεται από τη βιβλιοθήκη **ESP32-audioI2S-master** και υποστηρίζει τη ροή μορφών όπως MP3 απευθείας από απομακρυσμένες πηγές. Συμπληρωματικά, το αντικείμενο WiFiClientSecure επιτρέπει τη δημιουργία ασφαλών HTTPS συνδέσεων, απαραίτητων για την επικοινωνία με υπηρεσίες όπως Google Text-to-Speech ή Azure Cognitive Services (βλπ. Εικόνα 89).

```
153  //FLAGS FOR SYSTEM FUNCTIONALITY
154  int Flag=0 , SWITCH_ON_OFF = 0 , k , Only_once=0;
155  //-------------------------------------------------
156  // Create audio object
157  Audio audio;
158  WiFiClientSecure client;
```

Εικόνα 89:Δήλωση μεταβλητών (FLAGS)

Προτού προχωρήσουμε στην ανάλυση των βασικών συναρτήσεων void setup() και void loop(), κρίνεται αναγκαίο να εξετάσουμε τις επιμέρους συναρτήσεις που καλούνται στο εσωτερικό τους. Η κατανόηση αυτών των συναρτήσεων αποτελεί απαραίτητη προϋπόθεση για την ερμηνεία της συνολικής ροής του προγράμματος.

4.2.2 Ανάλυση βοηθητικών συναρτήσεων ESP32

Στη δεύτερη υποενότητα αναλύονται οι συναρτήσεις που καλούνται μέσα στις βασικές δομές του προγράμματος, δηλαδή στις setup() και loop(). Οι συναρτήσεις αυτές αποτελούν δομικά στοιχεία του αλγορίθμου, καθώς περιλαμβάνουν εξειδικευμένες λειτουργίες που επιτρέπουν την αρχικοποίηση, τον χειρισμό περιφερειακών μονάδων και την αλληλεπίδραση με εξωτερικές υπηρεσίες. Ενδεικτικά, περιλαμβάνονται συναρτήσεις για τη σύνδεση στο δίκτυο Wi-Fi, την εκκίνηση της οθόνης και του touch controller, τη ρύθμιση παραμέτρων του ήχου μέσω I2S, καθώς και την επικοινωνία με υπηρεσίες όπως το Google Text-to-Speech. Παρακάτω αναλύονται εκτενέστερα οι συναρτήσεις ως εξής:

- Η συνάρτηση **touch_calibrate()** είναι υπεύθυνη για τη βαθμονόμηση της οθόνης αφής, ώστε να εντοπίζεται σωστά το σημείο επαφής του χρήστη. Αρχικά, ελέγχει αν υπάρχει διαθέσιμο το σύστημα αρχείων SPIFFS και αν όχι, το διαμορφώνει και το ενεργοποιεί. Στη συνέχεια, ελέγχει αν υπάρχει αποθηκευμένο αρχείο βαθμονόμησης (CALIBRATION_FILE). Αν υπάρχει και δεν ζητείται επαναβαθμονόμηση (μέσω της μεταβλητής REPEAT_CAL), τότε διαβάζει τα αποθηκευμένα δεδομένα και τα εφαρμόζει στην οθόνη με την tft.setTouch(). Σε διαφορετική περίπτωση, εκτελείται νέα διαδικασία βαθμονόμησης, όπου ο χρήστης καλείται να αγγίξει καθορισμένα σημεία στην οθόνη. Μόλις ολοκληρωθεί η βαθμονόμηση, τα δεδομένα αποθηκεύονται στο SPIFFS για μελλοντική χρήση, αποφεύγοντας την ανάγκη επανάληψης της διαδικασίας κάθε φορά. Η διαδικασία αυτή διασφαλίζει τη σωστή αντιστοίχιση συντεταγμένων αφής με την οθόνη (βλπ. Εικόνα 90).

```
void touch_calibrate(){
  uint16_t calData[5];
  uint8_t calDataOK = 0;
  // check file system exists
  if (!SPIFFS.begin()){
    Serial.println("Formatting file system");
    SPIFFS.format();
    SPIFFS.begin();
  }
  // check if calibration file exists and size is correct
  if (SPIFFS.exists(CALIBRATION_FILE)){
    if (REPEAT_CAL){
      // Delete if we want to re-calibrate
      SPIFFS.remove(CALIBRATION_FILE);
    }
    else{
      File f = SPIFFS.open(CALIBRATION_FILE, "r");
      if (f){
        if (f.readBytes((char *)calData, 14) == 14)
          calDataOK = 1;
        f.close();
      }
    }
  }
  if (calDataOK && !REPEAT_CAL){
    // calibration data valid
    tft.setTouch(calData);
  }
  else{
    // data not valid so recalibrate
    tft.fillScreen(TFT_BLACK);
    tft.setCursor(20, 0);
    tft.setTextFont(2);
    tft.setTextSize(1);
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
    tft.println("Touch corners as indicated");
    tft.setTextFont(1);
    tft.println();
    if (REPEAT_CAL){
      tft.setTextColor(TFT_RED, TFT_BLACK);
      tft.println("Set REPEAT_CAL to false to stop this running again!");
    }
    tft.calibrateTouch(calData, TFT_MAGENTA, TFT_BLACK, 15);
    tft.setTextColor(TFT_GREEN, TFT_BLACK);
    tft.println("Calibration complete!");

    // store data
    File f = SPIFFS.open(CALIBRATION_FILE, "w");
    if (f){
      f.write((const unsigned char *)calData, 14);
      f.close();
    }
  }
}
```

Εικόνα 90:Συνάρτηση touch_calibrate

- Η συνάρτηση **ConnecttoWiFi()** έχει ως βασικό σκοπό τη σύνδεση του ESP32 σε ασύρματο δίκτυο Wi-Fi και τη διαχείριση της επικοινωνίας με την ESP32-CAM μέσω της σειριακής διεπαφής softSerial (βλπ. Εικόνα 91). Αρχικά, διαβάζει τις τιμές SSID και κωδικού πρόσβασης από τη μνήμη EEPROM και τις στέλνει στην ESP32-CAM για συγχρονισμό. Έπειτα ξεκινά η διαδικασία σύνδεσης στο Wi-Fi, με επαναλαμβανόμενο έλεγχο της κατάστασης μέχρι να επιβεβαιωθεί η σύνδεση. Αφού επιτευχθεί σύνδεση, εμφανίζεται σχετικό μήνυμα στην οθόνη TFT και ο χρήστης ενημερώνεται και με ηχητικό μήνυμα μέσω Google TTS. Η συνάρτηση παραμένει σε βρόχο, περιμένοντας είτε απάντηση από την ESP32-CAM είτε να παρέλθει συγκεκριμένος χρόνος αναμονής. Αν ληφθεί μήνυμα, αυτό εμφανίζεται στην οθόνη, και ενεργοποιείται η αναπαραγωγή του ηχητικού μηνύματος. Το watchdog reset (esp_task_wdt_reset) εξασφαλίζει την ομαλή λειτουργία του συστήματος

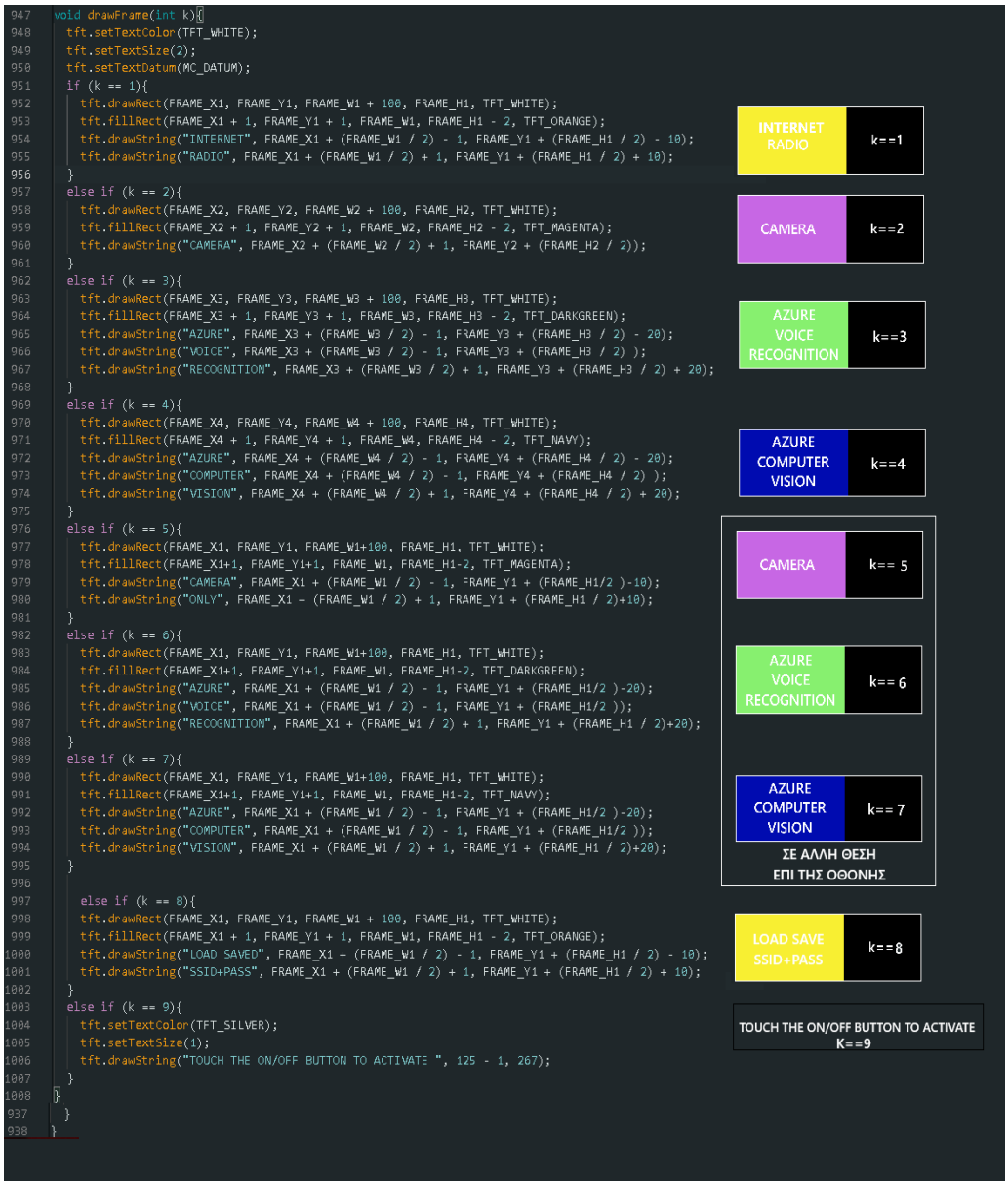
χωρίς να προκληθεί επανεκκίνηση λόγω καθυστέρησης, ενώ στο τέλος γίνεται έλεγχος χρονισμού για την ασφαλή έξοδο από τον βρόχο.

```
void ConnecttoWiFi(){
  const uint32_t timeout = 10000; // 10 second timeout
  const uint32_t speechDuration = 5000; // 5 second speech duration
  const uint32_t startTime = millis();
  bool speechPlayed = false;
  uint32_t speechStartTime = 0;

  String ssid = readStringFromEEPROM(SSID_ADDR) ;
  //softSerial.println(ssid); //SEND SSID TO ESP32 CAM
  String password = readStringFromEEPROM(PASS_ADDR) ;
  softSerial.println(ssid + "\t"+ password); //SEND PASSWORD TO ESP32 CAM
  Serial.print("\nConnecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
    esp_task_wdt_reset();  // Reset watchdog after init
  }
  tft.setTextColor(TFT_GREEN);
  tft.setTextSize(1);
  tft.setTextDatum(MC_DATUM);
  tft.drawString("WiFi connected. (ESP32 TFT)   ", 120, 280);
  Serial.println("\n\nWiFi connected. (ESP32 TFT)\n\n");
  while (true){
    esp_task_wdt_reset();  // Reset watchdog after init
    // Check for incoming data or timeout
    if (softSerial.available() || (millis() - startTime < timeout)){
      if (softSerial.available()){
        String response = softSerial.readStringUntil('\n');
        response.trim(); // Remove any extra whitespace
        // Only process if we got actual data
        if (response.length() > 0){
          Serial.println("\n\nReceived from ESP32 CAM: " + response + "\n");
          // Update display
          tft.setTextColor(TFT_GREEN, TFT_BLACK); // Set background color to avoid overdraw
          tft.setTextSize(1);
          tft.setTextDatum(MC_DATUM);
          tft.drawString(response +"   ", 120, 290);
          tft.fillRect(204, 284, 10, 10, TFT_BLACK);

          // Play speech if not already done
          if (!speechPlayed){
            audio.connecttospeech("WIFI CONNECTED", "en"); // Google TTS
            speechPlayed = true;
            speechStartTime = millis();
          }
        }
      }
      // Exit condition after speech has played for 5 seconds
      if (speechPlayed && (millis() - speechStartTime >= speechDuration)){
        esp_task_wdt_reset();  // Reset watchdog after init
        break;
      }
    }
    else{
      // Timeout occurred
      Serial.println("Timeout waiting for ESP32 CAM response");
      break;
    }
    // Small delay to prevent CPU hogging
    delay(10);
  }
}
```

Εικόνα 91:Συνάρτηση ConnecttoWiFi (ESP32)

- Η συνάρτηση **drawFrame(int k)** είναι υπεύθυνη για τη σχεδίαση διαφόρων πλαισίων στην οθόνη TFT, ανάλογα με την τιμή της παραμέτρου k (βλπ. Εικόνα 92). Κάθε τιμή του k αντιστοιχεί σε ένα διαφορετικό γραφικό πλαίσιο του συστήματος, όπως για k == 1 σχεδιάζει γραφικό πλαίσιο για "INTERNET RADIO", για k == 2 το γραφικό πλαίσιο για "CAMERA", για k == 3 το γραφικό πλαίσιο για "AZURE VOICE RECOGNITION" και για k == 4 το γραφικό πλαίσιο για "AZURE COMPUTER VISION". Για k == 5,6,7 σχεδιάζει τα ίδια πλαίσια αντίστοιχα, αλλά σε διαφορετικές θέσεις, για k == 8: το γραφικό πλαίσιο για "LOAD SAVED SSID+PASS"και τέλος για k == 9 προβάλει ένα μήνυμα καθοδήγησης (TOUCH THE ON/OFF BUTTON TO ACTIVATE) προς τον χρήστη για αλληλεπίδραση με το κουμπί αφής. Η λειτουργία βασίζεται σε χρωματική διαφοροποίηση, γεωμετρικά σχήματα

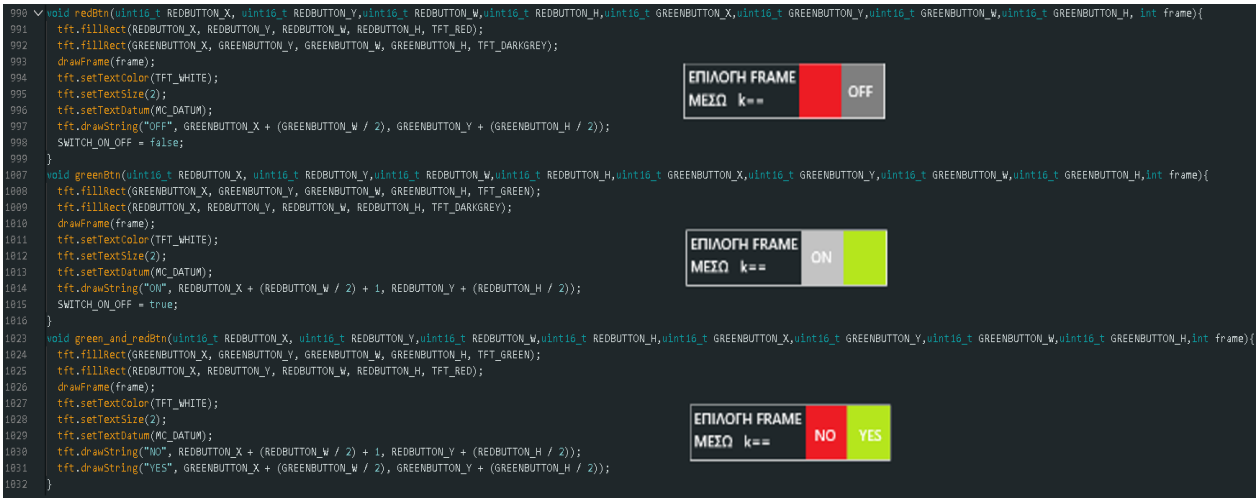
(drawRect, fillRect) και εμφανιζόμενο κείμενο (drawString) για να αποδώσει ένα απλό αλλά λειτουργικό γραφικό περιβάλλον χρήστη. Κάθε πλαίσιο λειτουργεί ως διαδραστικό στοιχείο στο περιβάλλον αφής.




```
void drawFrame(int k){
  tft.setTextColor(TFT_WHITE);
  tft.setTextSize(2);
  tft.setTextDatum(MC_DATUM);
  if (k == 1){
    tft.drawRect(FRAME_X1, FRAME_Y1, FRAME_W1 + 100, FRAME_H1, TFT_WHITE);
    tft.fillRect(FRAME_X1 + 1, FRAME_Y1 + 1, FRAME_W1, FRAME_H1 - 2, TFT_ORANGE);
    tft.drawString("INTERNET", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1 / 2) - 10);
    tft.drawString("RADIO", FRAME_X1 + (FRAME_W1 / 2) + 1, FRAME_Y1 + (FRAME_H1 / 2) + 10);
  }
  else if (k == 2){
    tft.drawRect(FRAME_X2, FRAME_Y2, FRAME_W2 + 100, FRAME_H2, TFT_WHITE);
    tft.fillRect(FRAME_X2 + 1, FRAME_Y2 + 1, FRAME_W2, FRAME_H2 - 2, TFT_MAGENTA);
    tft.drawString("CAMERA", FRAME_X2 + (FRAME_W2 / 2) + 1, FRAME_Y2 + (FRAME_H2 / 2));
  }
  else if (k == 3){
    tft.drawRect(FRAME_X3, FRAME_Y3, FRAME_W3 + 100, FRAME_H3, TFT_WHITE);
    tft.fillRect(FRAME_X3 + 1, FRAME_Y3 + 1, FRAME_W3, FRAME_H3 - 2, TFT_DARKGREEN);
    tft.drawString("AZURE", FRAME_X3 + (FRAME_W3 / 2) - 1, FRAME_Y3 + (FRAME_H3 / 2) - 20);
    tft.drawString("VOICE", FRAME_X3 + (FRAME_W3 / 2) - 1, FRAME_Y3 + (FRAME_H3 / 2) );
    tft.drawString("RECOGNITION", FRAME_X3 + (FRAME_W3 / 2) + 1, FRAME_Y3 + (FRAME_H3 / 2) + 20);
  }
  else if (k == 4){
    tft.drawRect(FRAME_X4, FRAME_Y4, FRAME_W4 + 100, FRAME_H4, TFT_WHITE);
    tft.fillRect(FRAME_X4 + 1, FRAME_Y4 + 1, FRAME_W4, FRAME_H4 - 2, TFT_NAVY);
    tft.drawString("AZURE", FRAME_X4 + (FRAME_W4 / 2) - 1, FRAME_Y4 + (FRAME_H4 / 2) - 20);
    tft.drawString("COMPUTER", FRAME_X4 + (FRAME_W4 / 2) - 1, FRAME_Y4 + (FRAME_H4 / 2) );
    tft.drawString("VISION", FRAME_X4 + (FRAME_W4 / 2) + 1, FRAME_Y4 + (FRAME_H4 / 2) + 20);
  }
  else if (k == 5){
    tft.drawRect(FRAME_X1, FRAME_Y1, FRAME_W1+100, FRAME_H1, TFT_WHITE);
    tft.fillRect(FRAME_X1+1, FRAME_Y1+1, FRAME_W1, FRAME_H1-2, TFT_MAGENTA);
    tft.drawString("CAMERA", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1/2 )-10);
    tft.drawString("ONLY", FRAME_X1 + (FRAME_W1 / 2) + 1, FRAME_Y1 + (FRAME_H1 / 2)+10);
  }
  else if (k == 6){
    tft.drawRect(FRAME_X1, FRAME_Y1, FRAME_W1+100, FRAME_H1, TFT_WHITE);
    tft.fillRect(FRAME_X1+1, FRAME_Y1+1, FRAME_W1, FRAME_H1-2, TFT_DARKGREEN);
    tft.drawString("AZURE", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1/2 )-20);
    tft.drawString("VOICE", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1/2 ));
    tft.drawString("RECOGNITION", FRAME_X1 + (FRAME_W1 / 2) + 1, FRAME_Y1 + (FRAME_H1 / 2)+20);
  }
  else if (k == 7){
    tft.drawRect(FRAME_X1, FRAME_Y1, FRAME_W1+100, FRAME_H1, TFT_WHITE);
    tft.fillRect(FRAME_X1+1, FRAME_Y1+1, FRAME_W1, FRAME_H1-2, TFT_NAVY);
    tft.drawString("AZURE", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1/2 )-20);
    tft.drawString("COMPUTER", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1/2 ));
    tft.drawString("VISION", FRAME_X1 + (FRAME_W1 / 2) + 1, FRAME_Y1 + (FRAME_H1 / 2)+20);
  }

    else if (k == 8){
    tft.drawRect(FRAME_X1, FRAME_Y1, FRAME_W1 + 100, FRAME_H1, TFT_WHITE);
    tft.fillRect(FRAME_X1 + 1, FRAME_Y1 + 1, FRAME_W1, FRAME_H1 - 2, TFT_ORANGE);
    tft.drawString("LOAD SAVED", FRAME_X1 + (FRAME_W1 / 2) - 1, FRAME_Y1 + (FRAME_H1 / 2) - 10);
    tft.drawString("SSID+PASS", FRAME_X1 + (FRAME_W1 / 2) + 1, FRAME_Y1 + (FRAME_H1 / 2) + 10);
  }
  else if (k == 9){
    tft.setTextColor(TFT_SILVER);
    tft.setTextSize(1);
    tft.drawString("TOUCH THE ON/OFF BUTTON TO ACTIVATE ", 125 - 1, 267);
  }
}
  }
}
```

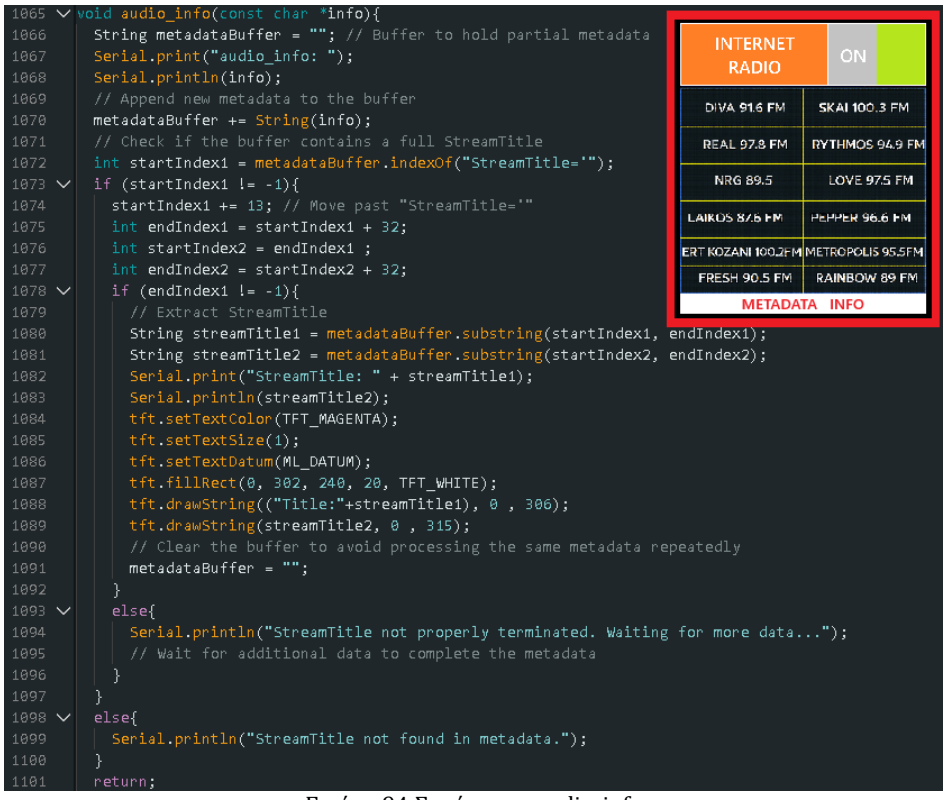
Εικόνα 92:Συνάρτηση drawFrame

- Οι συναρτήσεις **redBtn(), greenBtn()** και **green_and_redBtn()** είναι υπεύθυνες για την οπτική αναπαράσταση ενός διακόπτη ON/OFF και ενός διακόπτη NO/YES σε οθόνη TFT (βλπ. Εικόνα 93). Συγκεκριμένα, η redBtn() εμφανίζει ένα κόκκινο κουμπί και ένα γκρι κουμπί με την ένδειξη "OFF" απενεργοποιημένο, που δηλώνει την κατάσταση που βρίσκεται, ενώ η greenBtn() το αντίστροφο, δηλαδή εμφανίζει ένα πράσινο κουμπί και ένα γκρι κουμπί με την ένδειξη "ON" ενεργοποιημένο, που και αυτό δηλώνει την κατάσταση που βρίσκεται. Η τρίτη συνάρτηση green_and_redBtn() εμφανίζει ένα κόκκινο κουμπί και ένα πράσινο κουμπί με την ένδειξη "NO" και"YES" αντίστοιχα. Επιπλέον και οι τρεις συναρτήσεις, καλούν τη συνάρτηση drawFrame(frame) για να εμφανίσουν σχετικό γραφικό πλαίσιο. Η μεταβλητή SWITCH_ON_OFF, που αποτυπώνει την τρέχουσα κατάσταση του διακόπτη, ενημερώνεται μόνο από τις redBtn() και greenBtn().



Εικόνα 93:Συνάρτηση redBtn, greenBtn και green_and_redBtn

- Η συνάρτηση **audio_info()** είναι υπεύθυνη για την επεξεργασία και προβολή των μεταδεδομένων ήχου (όπως τίτλοι τραγουδιών) που λαμβάνονται μέσω ενός stream, όπως ένα Internet Radio (βλπ. Εικόνα 94).



Εικόνα 94:Συνάρτηση audio_info

- Η συνάρτηση **drawActivateButton()** σχεδιάζει ένα κυκλικό κόκκινο κουμπί με λευκό περίγραμμα και την ένδειξη "ACTIVATE" στο κέντρο (βλπ. Εικόνα 95). Το κουμπί αυτό είναι σχεδιασμένο για να διευκολύνεις την ενεργοποίηση λειτουργιών.

Εικόνα 95:Συνάρτηση drawActivateButton

- Η συνάρτηση **parseString_from_softSerial** λαμβάνει ως είσοδο μία μορφοποιημένη συμβολοσειρά (π.χ. 6/0.36@1024$1280?a grey surface with a black background%) και εξάγει τέσσερα βασικά στοιχεία: το ποσοστό εμπιστοσύνης (confidence), το ύψος (height), το πλάτος (width) και μία περιγραφή (description) (βλπ. Εικόνα 96). Τα στοιχεία αυτά αναλύονται με χρήση χαρακτήρων-οριοθετών (/, @, $, ?, %) και προβάλλονται στην οθόνη TFT, μαζί με την περιγραφή που διαχωρίζεται σε δύο γραμμές για καλύτερη εμφάνιση. Επιπλέον, κατά την ανάλυση εκτυπώνονται στο σειριακό monitor τα ενδιάμεσα αποτελέσματα για σκοπούς αποσφαλμάτωσης (debugging).

```
String parseString_from_softSerial(const String& softSerial_input){
  String confidenceStr , heightStr , widthStr , description , text1 , text2;
  float confidence = 0.0;
  int height = 0, width = 0;
  // Extract confidence (between '/' and '@')    6/0.36@1024$1280?a grey surface with a black background%
  int start_idx = softSerial_input.indexOf('/') + 1;
  int end_idx = softSerial_input.indexOf('@');
  if (start_idx > 0 && end_idx > start_idx) {
    confidenceStr = softSerial_input.substring(start_idx, end_idx);
    confidenceStr.trim();
    confidence = confidenceStr.toFloat();
    Serial.println("Confidence: " + confidenceStr); // Output for debugging
  }
  // Extract height (between '@' and '$')
  start_idx = softSerial_input.indexOf('@') + 1;
  end_idx = softSerial_input.indexOf('$');
  if (start_idx > 0 && end_idx > start_idx) {
    heightStr = softSerial_input.substring(start_idx, end_idx);
    heightStr.trim();
    height = heightStr.toInt();
    Serial.println("Height: " + heightStr); // Output for debugging
  }
  // Extract width (between '$' and '?')
  start_idx = softSerial_input.indexOf('$') + 1;
  end_idx = softSerial_input.indexOf('?');
  if (start_idx > 0 && end_idx > start_idx) {
    widthStr = softSerial_input.substring(start_idx, end_idx);
    widthStr.trim();
    width = widthStr.toInt();
    Serial.println("Width: " + widthStr); // Output for debugging
  }
  // Extract description (between '?' and '%')
  start_idx = softSerial_input.indexOf('?') + 1;
  end_idx = softSerial_input.indexOf('%');
  if (start_idx > 0 && end_idx > start_idx) {
    description = softSerial_input.substring(start_idx, end_idx);
    description.trim();
    Serial.println("Description: " + description); // Output for debugging
  }
  // Display on TFT screen
  tft.setTextColor(TFT_BLUE);
  tft.setTextSize(1);
  tft.setTextDatum(ML_DATUM);
  tft.drawString(confidenceStr,71,291);
  tft.drawString(heightStr,211,291);
  tft.drawString(widthStr,138,291);
  splitAfter_i_Spaces(description,text1,text2,6);
  tft.drawString(text2, 3 , 260);
  return description;
}
```

Εικόνα 96:Συνάρτηση parseString_from_softSerial

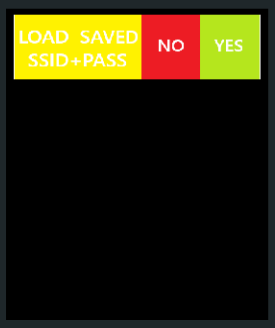
- Η συνάρτηση **splitAfter_i_Spaces** διαχωρίζει μια δοσμένη συμβολοσειρά σε δύο μέρη (text1 και text2), με βάση τον αριθμό των πρώτων k κενών διαστημάτων που συναντά. Διατρέχει χαρακτήρα προς χαρακτήρα την είσοδο, αυξάνει έναν μετρητή κάθε φορά που βρίσκει κενό, και προσθέτει τους χαρακτήρες είτε στο πρώτο είτε στο δεύτερο μέρος, ανάλογα με το αν έχει φτάσει τα k κενά. Τέλος, αφαιρεί ένα αρχικό κενό στο δεύτερο μέρος για καθαρότερο αποτέλεσμα στην εμφάνιση (βλπ. Εικόνα 97).

```
void splitAfter_i_Spaces(String input, String &text1, String &text2 ,int k){
  int spaceCount = 0;  // Counter for spaces
  text1 = "";          // Holds the first part
  text2 = "";          // Holds the second part
  // Iterate through the input string
  for (int i = 0; i < input.length(); i++){
    if (input[i] == ' '){
      spaceCount++;
    }
    // Append characters to the appropriate part
    if (spaceCount < k){
      text1 += input[i];
    }
    else{
      text2 += input[i];
    }
  }
  // Remove the leading space in part2 (optional)
  if (text2.startsWith(" ")){
    text2 = text2.substring(1);
  }
}
```

Εικόνα 97:Συνάρτηση splitAfter_i_Spaces

- Η συνάρτηση **intro_screen()** εμφανίζει στην αρχική οθόνη ένα μήνυμα, που μας προτρέπει τον χρήστη να επιλέξει αν θέλει να φορτώσει το αποθηκευμένο SSID και κωδικό (βλπ. Εικόνα 98). Συγκεκριμένα, καλεί τη συνάρτηση green_and_redBtn() για να εμφανίσει τα δύο κουμπιά επιλογής YES/NO. Έπειτα, εμφανίζει την τάση μέσω της voltage_checker() και στη συνέχεια, μπαίνει σε έναν βρόχο όπου ελέγχει συνεχώς αν έχει πατηθεί το πράσινο κουμπί (YES) ή το κόκκινο κουμπί (NO). Αν πατήθηκε το πράσινο, τότε διαβάζει και εμφανίζει το SSID και τον κωδικό από τη μνήμη EEPROM και θέτει την τιμή του k σε -1, δηλώνοντας ότι θα χρησιμοποιηθούν τα αποθηκευμένα στοιχεία. Αν πατήθηκε το κόκκινο, τότε εμφανίζει διαθέσιμα Wi-Fi δίκτυα με τη scanAndDrawWiFiNetworks() και θέτει την τιμή του k σε -2, δηλώνοντας ότι ο χρήστης θα εισάγει νέα στοιχεία σύνδεσης. Με αυτόν τον τρόπο, η intro_screen() καθοδηγεί τον χρήστη στην επιλογή μεταξύ αποθήκευσης ή εισαγωγής στοιχείων σύνδεσης πριν συνεχιστεί η λειτουργία της εφαρμογής.

```
1219  void intro_screen(){
1220    uint16_t x;
1221    uint16_t y;
1222    green_and_redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1, 8);
1223    voltage_checker();
1224    while (true) {
1225      esp_task_wdt_reset();  // Reset watchdog after init
1226      // Get touch coordinates once
1227      if (tft.getTouch(&x, &y)){
1228        // Check if the touch is within the GREEN Button area , load credentials (k==-1)
1229        if (x > GREENBUTTON_X1 && x < (GREENBUTTON_X1 + GREENBUTTON_W1) && y > GREENBUTTON_Y1 && y <= (GREENBUTTON_Y1 + GREENBUTTON_H1)){
1230          // Load SSID and Password from EEPROM
1231          String ssid = readStringFromEEPROM(SSID_ADDR);
1232          String pass = readStringFromEEPROM(PASS_ADDR); //PASS_ADDR
1233          tft.setTextDatum(MC_DATUM);
1234          tft.setTextSize(1);
1235          tft.setTextColor(TFT_RED);
1236          tft.drawString("---SSID---", 120 , 110);
1237          tft.setTextColor(TFT_WHITE);
1238          tft.drawString(ssid, 120 , 120);
1239          tft.setTextColor(TFT_RED);
1240          tft.drawString("---PASSWORD---", 120 , 130);
1241          tft.setTextColor(TFT_WHITE);
1242          tft.drawString(pass, 120 , 140);
1243          delay(3000);
1244          tft.fillScreen(TFT_BLACK);
1245          k=-1;
1246          break; // Exit the loop when the green button is pressed
1247        }
1248        // Check if the touch is within the RED Button area  , insert credentials (k==-2)
1249        if (x > REDBUTTON_X1 && x < (REDBUTTON_X1 + REDBUTTON_W1) && y > REDBUTTON_Y1 && y <= (REDBUTTON_Y1 + REDBUTTON_H1)){
1250          scanAndDrawWiFiNetworks();
1251          k=-2;
1252          break;  // Exit the loop when the red button is pressed
1253        }
1254      }
1255    }
1256  }
```



Εικόνα 98:Συνάρτηση intro_screen

- Η συνάρτηση **drawKey()** σχεδιάζει ένα πλήκτρο στην οθόνη TFT, λαμβάνοντας ως παραμέτρους τον χαρακτήρα του πλήκτρου, τις συντεταγμένες (x, y), καθώς και το πλάτος και ύψος του (βλπ. Εικόνα 99). Αρχικά, σχεδιάζει το περίγραμμα του πλήκτρου με λευκό χρώμα χρησιμοποιώντας την drawRect, και στη συνέχεια εμφανίζει τον χαρακτήρα στο εσωτερικό του πλήκτρου. Με την κατάλληλη ρύθμιση του setTextDatum(MC_DATUM) και τοποθέτηση του κειμένου στο κέντρο του ορθογωνίου, εξασφαλίζει ότι ο χαρακτήρας είναι πλήρως κεντραρισμένος τόσο οριζόντια όσο και κάθετα, προσφέροντας έτσι μια αισθητικά ευθυγραμμισμένη και ευανάγνωστη εμφάνιση για κάθε κουμπί του πληκτρολογίου.

```
1306  void drawKey(char key, int x, int y, int w, int h){
1307    tft.drawRect(x, y, w, h, TFT_WHITE);
1308    tft.setCursor(x + 10, y + 10);
1309    tft.print(key);
1310  }
```

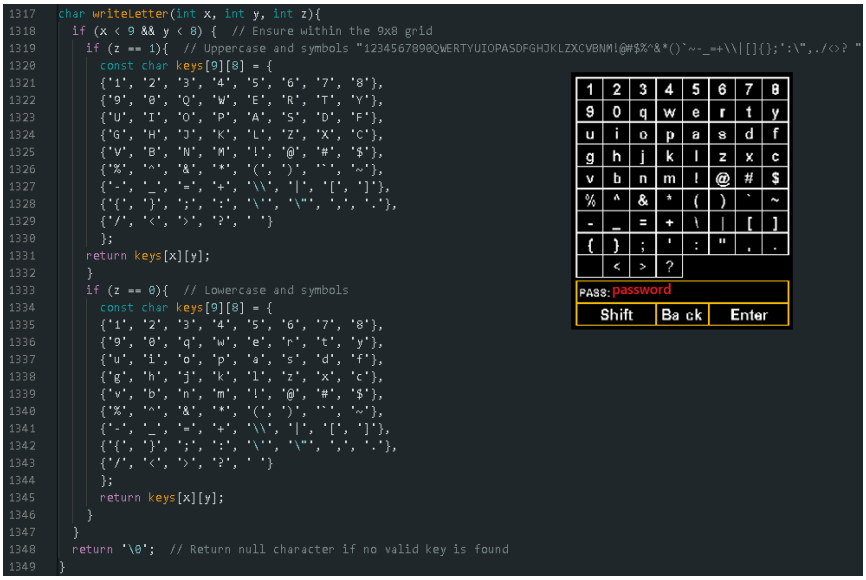
Εικόνα 99:Συνάρτηση drawKey

- Η συνάρτηση **showKeyboard()** εμφανίζει ένα εικονικό πληκτρολόγιο σε οθόνη TFT, επιτρέποντας την εναλλαγή μεταξύ κεφαλαίων και πεζών χαρακτήρων με βάση την τιμή της παραμέτρου a (βλπ. Εικόνα 100). Περιλαμβάνει έναν πίνακα χαρακτήρων για κάθε

μορφή (κεφαλαία και πεζά) και σχεδιάζει τα πλήκτρα σε διατάξεις 30x30 pixel με τρόπο που ταιριάζει στην οθόνη, χρησιμοποιώντας αυτόματο πέρασμα σε νέα γραμμή όταν γεμίσει οριζόντια η επιφάνεια. Η συνάρτηση καθαρίζει πρώτα την οθόνη, κατόπιν σχεδιάζει τα πλήκτρα και στο κάτω μέρος προσθέτει τρία ειδικά κουμπιά: Shift, Backspace και Enter, καθώς και ένα πλαίσιο για την είσοδο κωδικού με ετικέτα "PASS:"

Εικόνα 100:Συνάρτηση showKeyboard

- Η συνάρτηση **writeLetter()** επιστρέφει τον χαρακτήρα που βρίσκεται σε συγκεκριμένη θέση ενός εικονικού πληκτρολογίου τύπου πλέγματος 9x8, με βάση τις παραμέτρους x (στήλη), y (γραμμή) και z (λειτουργία αλλαγής πεζών/κεφαλαίων) (βλπ. Εικόνα 101). Αν η θέση είναι εντός ορίων και το z είναι 1, επιστρέφει τον αντίστοιχο χαρακτήρα από τον πίνακα με κεφαλαία γράμματα και σύμβολα. Αν το z είναι 0, επιστρέφει τον χαρακτήρα από τον πίνακα με πεζά γράμματα και σύμβολα. Οι χαρακτήρες είναι αποθηκευμένοι σε σταθερούς πίνακες 9x8, και η συνάρτηση επιστρέφει τον χαρακτήρα από τη θέση keys[x][y]. Αν η θέση είναι εκτός ορίων ή η τιμή του z δεν είναι 0 ή 1, επιστρέφει τον null χαρακτήρα '\0'.



Εικόνα 101:Συνάρτηση writeLetter

- Οι δύο συναρτήσεις **readStringFromEEPROM()** και **writeStringToEEPROM()** χειρίζονται την αποθήκευση και ανάγνωση αλφαριθμητικών τιμών (τύπου String) στη μνήμη EEPROM(βλπ. Εικόνα 102). Η writeStringToEEPROM γράφει κάθε χαρακτήρα της συμβολοσειράς στη μνήμη ξεκινώντας από μια δοσμένη διεύθυνση και προσθέτει στο τέλος ένα null-terminator ('\0') για να σηματοδοτήσει το τέλος της συμβολοσειράς, ενώ στη συνέχεια καλεί την EEPROM.commit() για να εξασφαλίσει την αποθήκευση. Αντίστοιχα, η readStringFromEEPROM ανακτά τα αποθηκευμένα δεδομένα διαβάζοντας χαρακτήρες από την EEPROM μέχρι να εντοπίσει τον null-terminator και επιστρέφει τη συμβολοσειρά που δημιουργήθηκε. Αυτές οι συναρτήσεις είναι χρήσιμες για την αποθήκευση ρυθμίσεων, όπως SSID Wi-Fi, που πρέπει να διατηρούνται μετά από επανεκκίνηση της συσκευής.

```
1474  String readStringFromEEPROM(int startAddr){
1475    String result = "";
1476    char ch;
1477    int index = 0;
1478
1479    // Read characters until we encounter a null-terminator
1480    while (true){
1481      ch = EEPROM.read(startAddr + index);
1482      if (ch == '\0') break;  // Stop when null-terminator is encountered
1483      result += ch;  // Append the character to the string
1484      index++;
1485    }
1486    return result;  // Return the reconstructed string
1487  }
1488  //-------------------------------------------------------------------------------------
1494  void writeStringToEEPROM(String str, int startAddr){
1495    for (int i = 0; i < str.length(); i++){
1496      EEPROM.write(startAddr + i, str[i]);  // Write each character to EEPROM
1497    }
1498    EEPROM.write(startAddr + str.length(), '\0');  // Add null-terminator at the end
1499    EEPROM.commit();  // Commit changes to EEPROM
1500  }
```

Εικόνα 102:Συναρτήσεις readStringFromEEPROM και writeStringFromEEPROM

- Η συνάρτηση **voltage_checker()** μετρά την τάση μέσω ενός διαιρέτη τάσης και την εμφανίζει στην οθόνη. Αρχικά, ενεργοποιεί τον διαιρέτη τάσης μέσω του CONTROL_PIN και περιμένει 50ms για σταθεροποίηση. Έπειτα διαβάζει την αναλογική τιμή από το VOLTAGE_PIN, τη μετατρέπει σε τάση βασιζόμενος στην αναλογία 8.71 (πιθανώς αποτέλεσμα του διαιρέτη τάσης και της αναφοράς του ADC) και εμφανίζει το αποτέλεσμα τόσο στη σειριακή θύρα όσο και στην οθόνη TFT. Τέλος, απενεργοποιεί τον αισθητήρα. Η οθόνη ενημερώνεται δυναμικά καθαρίζοντας το προηγούμενο αποτέλεσμα και γράφοντας τη νέα τάση με δύο δεκαδικά ψηφία (βλπ. Εικόνα 103).

```
1520  void voltage_checker(){
1521  // Enables voltage divider
1522    digitalWrite(CONTROL_PIN, HIGH);
1523    delay(50); // stabilization time
1524
1525    // Read ADC value
1526    int raw = analogRead(VOLTAGE_PIN);
1527
1528    // Convert to voltage
1529    float voltage = (raw / 4095.0) * 8.71;
1530
1531    Serial.print("Raw ADC: ");
1532    Serial.print(raw);
1533    Serial.print("  -> Voltage: ");
1534    Serial.print(voltage, 2);
1535    Serial.println(" V");
1536
1537    // Turn off sensor
1538    digitalWrite(CONTROL_PIN, LOW);
1539    tft.setTextColor(TFT_WHITE);
1540    tft.setTextSize(1);
1541    tft.setTextDatum(MC_DATUM);
1542    String voltageStr = String(voltage, 2); // Convert float to String with 2 decimal places
1543    tft.fillRect(195, 3, 40, 12, TFT_BLACK);
1544    tft.drawString(voltageStr + "v", 215, 9 );
1545  }
```

Εικόνα 103:Συνάρτηση voltage_checker

- Η συνάρτηση **scanAndDrawWiFiNetworks()**εκτελεί σάρωση διαθέσιμων δικτύων Wi-Fi, τα εμφανίζει σε μια οθόνη TFT και επιτρέπει στον χρήστη να επιλέξει κάποιο από αυτά μέσω αφής, ώστε να αποθηκευτεί στη μνήμη EEPROM και να σταλεί μέσω UART σε άλλη συσκευή (βλπ. Εικόνα 104). Αρχικά καθαρίζει την οθόνη και εμφανίζει τα SSID των δικτύων

που εντοπίζονται. Αν βρεθούν περισσότερα από 8, εμφανίζει δύο σειρές πλήκτρων επιλογής (με αριθμούς), ενώ αν είναι 8 ή λιγότερα, χρησιμοποιείται μόνο μία σειρά. Ο χρήστης μπορεί να επιλέξει SSID πατώντας το αντίστοιχο κουμπί στην οθόνη αφής. Η επιλεγμένη τιμή αποθηκεύεται σε EEPROM και προβάλλεται στην οθόνη για επιβεβαίωση. Επιπλέον, γίνεται χρήση watchdog reset για αποφυγή επανεκκίνησης κατά τη διάρκεια μεγάλων λειτουργιών.

```
void scanAndDrawWiFiNetworks(){
  uint16_t a,b;
  int num;
  tft.fillScreen(TFT_BLACK);  // Clear the screen
  tft.setTextColor(TFT_WHITE);
  tft.setTextSize(1);
  // Scan for Wi-Fi networks
  int maxSSIDs = WiFi.scanNetworks();
  esp_task_wdt_reset();
  String* ssidList = new String[maxSSIDs];
  //String  ssidList[maxSSIDs];  // Array to store SSIDs
  Serial.println("SSIDS=" + String(maxSSIDs));
  if (maxSSIDs == 0){
    tft.setCursor(10, 10);
    tft.println("No networks found.");
    return;
  }
  // Display the list of Wi-Fi networks
  tft.setCursor(65, 10);
  tft.println("Available Networks:");
  if(maxSSIDs!=0){
    int y = 30;  // Starting position for network names
    for (int i = 0; i < maxSSIDs; i++){
      if (y > tft.height() - 10){  // Stop if out of scr
        tft.setCursor(10, y);
        ssidList[i]=WiFi.SSID(i);
        tft.println("...more");
        break;
      }
      // Display SSID
      String ssid = WiFi.SSID(i);
      ssidList[i]=WiFi.SSID(i);
      Serial.println(String(i+1) + "."+ ssidList[i]);
      tft.setCursor(10, y);
      tft.printf("%d: %s", i + 1, ssid.c_str());
      y += 12;  // Move to the next line
      esp_task_wdt_reset();
    }
  }
  if (maxSSIDs > 8){
    Serial.println("SSIDS>8: " + String(maxSSIDs));
    for (int i = 0; i < 8; i++){
      tft.drawRect(30*i, 260, 30, 30, TFT_WHITE);
      tft.setTextSize(2);
      tft.drawString(String((i+1)), i*30+15 , 275);
    }
    for (int i = 8; i < maxSSIDs; i++){
      tft.drawRect((30*i-8), 290, 30, 30, TFT_WHITE);
      tft.drawString(String((i+1)), (i-7)*30+15 , 305);
    }
    while (true){
      esp_task_wdt_reset();
      // Get touch coordinates once
      if (tft.getTouch(&a, &b)){
        // Check if the touch is within the Green Button area , load gredentials
        if (a > 0 && a < 240 && b > 260 && b <= 290){
          num=a/30;
          Serial.println("s1>8: "+String(num));
        }
        if (a > 0 && a < (maxSSIDs-8)*30 && b > 290 && b <= 320){
          num=(a/30)+8;
          Serial.println("s2>8: "+String(num));
        }
        // Save SSID and Password from EEPROM
        writeStringToEEPROM( ssidList[num], SSID_ADDR);
        softSerial.println(ssidList[num]); //SEND SSID TO ESP32 CAM BY UART
        tft.setTextDatum(MC_DATUM);
        tft.setTextSize(1);
        tft.setTextColor(TFT_RED);
        tft.drawString("---SSID---", 120 , 110);
        tft.setTextColor(TFT_WHITE);
        tft.drawString(ssidList[num], 120 ,120);
        Serial.println("ssidlist no=" + ssidList[num]);
        delay(3000);
        break; // Exit the loop when the red button is pressed
      }
    }
  }
  if (maxSSIDs<=8){
    Serial.println("ssids<8:"+String(maxSSIDs));
    for (int i = 0; i < maxSSIDs; i++){
      tft.drawRect(30*i, 290, 30, 30, TFT_WHITE);
      tft.setTextSize(2);
      tft.drawString(String((i+1)), i*30+15 ,305);
    }
    while (true){
      esp_task_wdt_reset();
      // Get touch coordinates once
      if (tft.getTouch(&a, &b)){
        // Check if the touch is within the Green Button area , save SSID
        if (a > 0 && a < 30*maxSSIDs && b > 290 && b <= 320){
          num=a/30;
          Serial.println("s<=8: (" + String(num) + ")" );
        }
        // Save SSID  to EEPROM
        writeStringToEEPROM(ssidList[num], SSID_ADDR);
        softSerial.println(ssidList[num]);  //SEND SSID TO ESP32 CAM BY UART
        tft.fillScreen(TFT_BLACK);
        tft.setTextDatum(MC_DATUM);
        tft.setTextSize(1);
        tft.setTextColor(TFT_RED);
        tft.drawString("---SSID---", 120 , 110);
        tft.setTextColor(TFT_MAGENTA);
        tft.drawString(ssidList[num], 120 ,120);
        Serial.println("SSID List No=" + ssidList[num]);
        delay(3000);
        break; // Exit the loop when the red button is pressed
      }
    }
  }
  delete[] ssidList;
}
```

Εικόνα 104:Συνάρτηση scanAndDrawWiFiNetworks

- Η συνάρτηση **generateSpeech_With_GoogleTTL()** χρησιμοποιεί την υπηρεσία Google Text-to-Speech (TTS) για να μετατρέψει ένα κείμενο (String) σε φωνή. Αρχικά ελέγχει αν η συσκευή είναι συνδεδεμένη στο WiFi, αν δεν είναι, εκτυπώνει μήνυμα σφάλματος και τερματίζει την εκτέλεση. Αν υπάρχει σύνδεση, καλεί τη μέθοδο connecttospeech από το αντικείμενο audio, μετατρέποντας το κείμενο σε φωνή χρησιμοποιώντας την ελληνική γλώσσα ("el-GR"). Η μετατροπή του String σε const char* είναι απαραίτητη για συμβατότητα με τη μέθοδο του TTS API (βλπ. Εικόνα 105).

```
1497  void generateSpeech_With_GoogleTTL(const String& text) {
1498      if (WiFi.status() != WL_CONNECTED) {
1499          Serial.println("WiFi not connected");
1500          return;  // Exit function if no WiFi
1501      }
1502
1503      //audio.connecttospeech(text.c_str(), "en"); // Convert String to const char*
1504      audio.connecttospeech(text.c_str(), "el-GR"); // Convert String to const char*
1505  }
```

Εικόνα 105:Συνάρτηση generateSpeech_With_GoogleTTL

- Η συνάρτηση **translateText()** έχει ως σκοπό τη μετάφραση αγγλικού κειμένου στα ελληνικά με τη βοήθεια της υπηρεσίας **Azure Translator**. Αρχικά, γίνεται σύνδεση με το κατάλληλο **endpoint** του Translator API, παρέχοντας το απαραίτητο **κλειδί πρόσβασης** και την **περιοχή λειτουργίας**. Το κείμενο προς μετάφραση αποστέλλεται σε μορφή **JSON** μέσω ενός **HTTP POST** αιτήματος. Εάν η απόκριση είναι επιτυχής (κωδικός 200), το μεταφρασμένο κείμενο εξάγεται από τη **JSON απάντηση**.

  Ωστόσο, δεν είναι δυνατή η απεικόνιση του μεταφρασμένου ελληνικού κειμένου στην οθόνη TFT, καθώς η βιβλιοθήκη γραφικών που χρησιμοποιείται δεν υποστηρίζει ελληνικούς χαρακτήρες. Παρ' όλα αυτά, η μετάφραση αξιοποιείται για **παραγωγή ομιλίας** μέσω της συνάρτησης **Google TTS**, η οποία επιτρέπει την ακουστική απόδοση του ελληνικού κειμένου. Η υλοποίηση αυτή επιτρέπει την αυτόματη και άμεση απόδοση των περιγραφών του συστήματος στα ελληνικά μέσω ήχου, διευκολύνοντας τη χρήση από ελληνόφωνους χρήστες (βλπ. Εικόνα 106).

```
1553  String translateText(const String& input){
1554    const String key    = "8os7NrRGfBIC19IgoQ6T7us1KpzCozwQV5FJn9K1vAtdXq6Ofk6DJQQJ99BFAC5RqLJXJ3w3AAAbACOGvUVv";
1555    const String region  = "westeurope";
1556    HTTPClient http;
1557    String text1, text2;
1558    String url = "https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&from=en&to=el";
1559    esp_task_wdt_reset();  // Reset watchdog after init
1560    http.begin(url);
1561    http.addHeader("Ocp-Apim-Subscription-Key", key);
1562    http.addHeader("Ocp-Apim-Subscription-Region", region);
1563    http.addHeader("Content-Type", "application/json");
1564
1565    String body = "[{\"Text\":\"" + input + "\"}]";
1566    int code = http.POST(body);
1567
1568    if (code == 200){
1569      String resp = http.getString();
1570      esp_task_wdt_reset();  // Reset watchdog after init
1571      int p1 = resp.indexOf("\"text\":\"") + 8;
1572      int p2 = resp.indexOf("\"", p1);
1573      if (p1 >= 8 && p2 > p1) {
1574        String translated = resp.substring(p1, p2);
1575        http.end();
1576        esp_task_wdt_reset();  // Reset watchdog after init
1577        return translated;
1578      }
1579      else{
1580        http.end();
1581        esp_task_wdt_reset();  // Reset watchdog after init
1582        return "Error: Parsing failed";
1583      }
1584    }
1585    else{
1586      http.end();
1587      esp_task_wdt_reset();  // Reset watchdog after init
1588      return "Error: " + String(code);
1589    }
1590  }
```

Εικόνα 106:Συνάρτηση translateText

4.2.3 Ανάλυση βασικών συναρτήσεων setup() και loop() του ESP32

Αφού εξηγήθηκαν αναλυτικά οι επιμέρους συναρτήσεις, ακολουθεί η παρουσίαση του βασικού κορμού του προγράμματος, ο οποίος περιλαμβάνεται στις συναρτήσεις setup() και loop(). Οι δύο αυτές συναρτήσεις συνθέτουν τη θεμελιώδη δομή κάθε εφαρμογής στο περιβάλλον του Arduino. Η setup() εκτελείται μία φορά κατά την εκκίνηση του συστήματος και χρησιμοποιείται για τις αρχικές ρυθμίσεις, ενώ η loop() περιέχει τον επαναλαμβανόμενο κώδικα που εκτελείται συνεχώς όσο η συσκευή παραμένει ενεργή. Στη συνέχεια, θα αναλυθεί λεπτομερώς το περιεχόμενο αυτών των δύο συναρτήσεων, με σκοπό την πλήρη κατανόηση της ροής του προγράμματος και της συνεργασίας μεταξύ των επιμέρους υπομονάδων.

Η συνάρτηση **setup()** αρχικοποιεί βασικά υποσυστήματα του ESP και καλεί συναρτήσεις, όπως:

- **Watchdog Timer:** Αναλύεται στην υποενότητα 4.1.3 Σελίδα 84
- **EEPROM:** Γίνεται η αποθήκευση των δεδομένων στο SPIFFS
- **Σειριακή Επικοινωνία:** Αναλύεται στην υποενότητα 4.1.3 Σελίδα 84
- **ADC (Analog-to-Digital Converter):** Η αντίστοιχη έξοδος (CONTROL_PIN) ορίζεται σε χαμηλή κατάσταση (LOW). Και ορίζεται η ανάλυση στα 12 bits για ακριβέστερες αναγνώσεις.
- **Οθόνη TFT:** αρχικοποιείται η οθόνη, ρυθμίζεται η περιστροφή της, γίνεται βαθμονόμηση αφής και τέλος καθαρίζεται η οθόνη με μαύρο χρώμα, προετοιμάζοντας το σύστημα για λειτουργία.
- η συνάρτηση **intro_screen()** χρησιμοποιείται για την εμφάνιση μιας εισαγωγικής οθόνης. Η συνάρτηση αναλύεται στην παραπάνω υποενότητα 4.2.2 Σελίδα 97.
- η συνάρτηση **esp_task_wdt_reset()**, η οποία μηδενίζει τον χρονομετρητή επιτήρησης εργασιών (watchdog timer) (βλπ. Εικόνα 107).

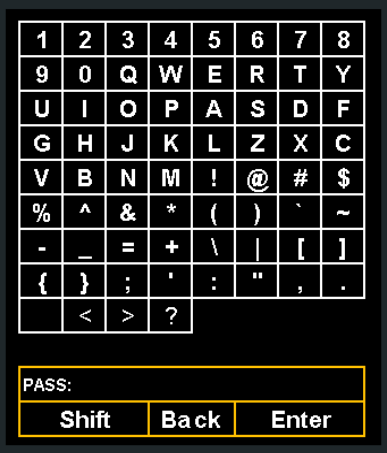
```
174  void setup(void){
175    // Initialize Watchdog Timer with proper configuration
176    esp_task_wdt_config_t wdt_config = {
177      .timeout_ms = WDT_TIMEOUT_SECONDS * 1000,
178      .idle_core_mask = (1 << portNUM_PROCESSORS) - 1, // Watch all cores
179      .trigger_panic = true
180    };
181    esp_task_wdt_init(&wdt_config);
182    esp_task_wdt_add(NULL);  // Add main task to watchdog
183      // Initialize EEPROM
184    EEPROM.begin(EEPROM_SIZE);
185    Serial.begin(115200);        // Set up hardwareSerial_0(USB) at 115200 baud
186    softSerial.begin(9600);      // Set up SoftwareSerial at 9600 baud
187    pinMode(CONTROL_PIN, OUTPUT);
188    digitalWrite(CONTROL_PIN, LOW);  // Initially off
189    analogReadResolution(12); // 0 - 4095  // Configure ADC
190      tft.init();
191    // Set the rotation before we calibrate
192    tft.setRotation(2);
193    // call screen calibration
194    touch_calibrate();
195    // clear screen
196    tft.fillScreen(TFT_BLACK);
197    intro_screen();
198    esp_task_wdt_reset();  // Reset watchdog after init
```

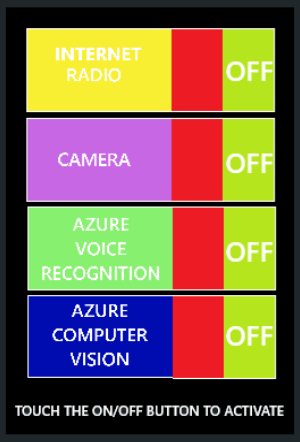
Εικόνα 107:Κώδικας εντός συνάρτησης setup() 1 - WDT, EEPROM, ADC, TFT Init-config

Αυτό το τμήμα κώδικα που απεικονίζεται στην Εικόνα 108 υλοποιεί μια διαδραστική διεπαφή για την εισαγωγή κωδικού πρόσβασης μέσω οθόνης αφής σε ένα σύστημα βασισμένο σε ESP32. Όταν η τιμή της μεταβλητής k είναι -2, εμφανίζεται ένα εικονικό πληκτρολόγιο με δυνατότητα εναλλαγής πεζών-κεφαλαίων (SHIFT), διαγραφής χαρακτήρων (BACKSPACE) και επιβεβαίωσης εισόδου (ENTER). Ο χρήστης εισάγει τον κωδικό αγγίζοντας τα αντίστοιχα πλήκτρα, και κάθε χαρακτήρας εμφανίζεται στην οθόνη, ενώ ταυτόχρονα ενημερώνεται η συμβολοσειρά temp. Ο watchdog timer επαναφέρεται συνεχώς για την αποφυγή ανεπιθύμητης επανεκκίνησης κατά την εισαγωγή. Όταν πατηθεί ENTER, ο κωδικός

αποθηκεύεται στη μνήμη EEPROM και το σύστημα καθαρίζει την οθόνη, σχεδιάζει κουμπιά ελέγχου, και προχωρά σε σύνδεση με το δίκτυο Wi-Fi.

```
if (k == -2){ // Insert the credentials (password)
  showKeyboard(1);
  temp = "";
  k = 0;
  Serial.printf("Free Heap: %d bytes\n", ESP.getFreeHeap());
  //uint16_t currentIndex = 0; // Declared outside the loop to preserve the value
  while (true){
    esp_task_wdt_reset();  // Reset watchdog after init
    uint16_t a = 0, b = 0; // Initialize touch coordinates
    uint16_t col, row;
    // Get touch coordinates once
    if (tft.getTouch(&a, &b)){
      if (a > 0 && a < 80 && b > 290 && b <= 320){ // SHIFT PRESSED
        showKeyboard((k % 2)); // Toggle keyboard case
        k++;
        Serial.println("SHIFT PRESSED");
        delay(300);
      }
      // BACKSPACE PRESSED
      if (a > 81 && a < 160 && b > 290 && b <= 320){
        if (!temp.isEmpty()){ // Check if password is not empty
          temp.remove(temp.length() - 1); // Remove the last character
        }
        k=k-10;
        if (k<=0){
          tft.fillRect(35, 275, 10, 10, TFT_BLACK);
          k=0;
        }
        if (k>0){
          tft.fillRect(35+k, 275, 10, 10, TFT_BLACK);
        }
        Serial.println("BACKSPACE PRESSED");
        delay(300);
      }
      if (a > 161 && a < 240 && b > 290 && b <= 320){ // ENTER PRESSED
        Serial.println("ENTER PRESSED");
        delay(300);
        Serial.println("Password: " + temp);

        writeStringToEEPROM(temp, PASS_ADDR); // Save password to EEPROM
        //softSerial.println(temp); //SEND PASSWORD TO ESP32 CAM
        // Clear the screen
        tft.fillScreen(TFT_BLACK);
        break; // Exit the loop when ENTER is pressed
      }
      // KEYBOARD PRESSED
      if ((a > 0 && a < 240 && b > 0 && b <= 240) || (a > 0 && a < 150 && b > 241 && b <= 270)){
        row = b / 30;
        col = a / 30;
        // Get the letter corresponding to the keypress
        char letter = writeLetter(row, col, (k % 2));
        Serial.printf("Letter: %c (Row: %d, Col: %d)\n", letter, row, col);
        tft.setCursor(35+k, 275);
        tft.setTextSize(1);
        tft.printf("%s ", String(letter).c_str());
        temp.concat(letter);
        //temp += letter; // Append the letter to the password string
        Serial.println(temp); // Debug output for the current password
        Serial.printf("Password: %s (k: %d)\n", temp.c_str(), k);
        delay(200); // Reduced delay for better responsiveness
        k=k+10;
      }
    }
  }
  tft.fillScreen(TFT_BLACK);
  redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1, 1);
  redBtn(REDBUTTON_X2, REDBUTTON_Y2, REDBUTTON_W2, REDBUTTON_H2,GREENBUTTON_X2, GREENBUTTON_Y2, GREENBUTTON_W2, GREENBUTTON_H2, 2);
  redBtn(REDBUTTON_X3, REDBUTTON_Y3, REDBUTTON_W3, REDBUTTON_H3,GREENBUTTON_X3, GREENBUTTON_Y3, GREENBUTTON_W3, GREENBUTTON_H3, 3);
  redBtn(REDBUTTON_X4, REDBUTTON_Y4, REDBUTTON_W4, REDBUTTON_H4,GREENBUTTON_X4, GREENBUTTON_Y4, GREENBUTTON_W4, GREENBUTTON_H4, 4);
  drawFrame(9);
  ConnecttoWiFi();    // Connect to Wi-Fi
}
```



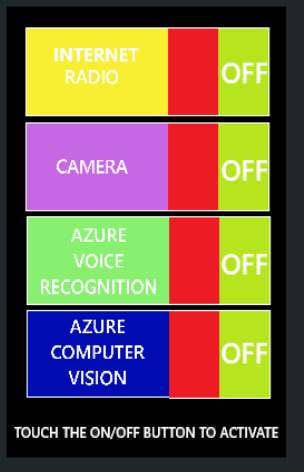


Εικόνα 108:Κώδικας εντός συνάρτησης setup() 2 - Insert WIFI credentials

Αυτό το κομμάτι κώδικα εκτελείται όταν η μεταβλητή k έχει τιμή -1, και χρησιμοποιείται για την αυτόματη φόρτωση των αποθηκευμένων διαπιστευτηρίων (credentials) από τη μνήμη EEPROM (βλπ. Εικόνα 109). Αρχικά καθαρίζεται η οθόνη και δημιουργούνται τέσσερα διαδραστικά κουμπιά μέσω της redBtn, καθώς και ένα πλαίσιο κειμένου με τη συνάρτηση drawFrame(9). Στη συνέχεια, καλείται η ConnecttoWiFi() για σύνδεση στο ασύρματο δίκτυο

χρησιμοποιώντας τα αποθηκευμένα στοιχεία. Επίσης ορίζεται η σύνδεση του ήχου με το I2S ενισχυτή MAX98357 μέσω της audio.setPinout(), και τέλος γίνεται επαναφορά του watchdog timer με esp_task_wdt_reset() για την αποφυγή ανεπιθύμητης επανεκκίνησης.

```
273   if (k==-1){//load the grentantials from EEPROM
274     tft.fillScreen(TFT_BLACK);
275     redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1, 1);
276     redBtn(REDBUTTON_X2, REDBUTTON_Y2, REDBUTTON_W2, REDBUTTON_H2,GREENBUTTON_X2, GREENBUTTON_Y2, GREENBUTTON_W2, GREENBUTTON_H2, 2);
277     redBtn(REDBUTTON_X3, REDBUTTON_Y3, REDBUTTON_W3, REDBUTTON_H3,GREENBUTTON_X3, GREENBUTTON_Y3, GREENBUTTON_W3, GREENBUTTON_H3, 3);
278     redBtn(REDBUTTON_X4, REDBUTTON_Y4, REDBUTTON_W4, REDBUTTON_H4,GREENBUTTON_X4, GREENBUTTON_Y4, GREENBUTTON_W4, GREENBUTTON_H4, 4);
279     drawFrame(9);
280     // Connect to Wi-Fi
281     ConnecttoWiFi();
282   }
283   client.setInsecure();  // Disable SSL certificate validation (use only if needed)
284  // Connect MAX98357 I2S Amplifier Module
285   audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
286
287   esp_task_wdt_reset();  // Reset watchdog after init
288 }
289
```



Εικόνα 109:Κώδικας εντός συνάρτησης setup() 3 - Load WIFI credentials

Loop:Αυτό το τμήμα του κώδικα εκτελεί περιοδικό έλεγχο της τάσης του συστήματος σε καθορισμένα χρονικά διαστήματα. Χρησιμοποιεί τη συνάρτηση millis() για να υπολογίσει τον χρόνο που πέρασε από την τελευταία μέτρηση (lastMeasureTime) και, όταν αυτός ξεπεράσει το προκαθορισμένο διάστημα (MEASURE_INTERVAL), καλεί τη συνάρτηση voltage_checker() για να πραγματοποιηθεί νέα μέτρηση. Μετά από κάθε έλεγχο (ή ακόμα κι αν δεν έγινε), γίνεται επαναφορά του watchdog timer με τη esp_task_wdt_reset() ώστε να διασφαλιστεί ότι το σύστημα δεν θα επανεκκινηθεί λόγω καθυστέρησης ή παγώματος στη διαδικασία μέτρησης (βλπ. Εικόνα 110).

```
302  void loop(){
303    unsigned long currentTime = millis();
304    if (cusrrentTime - lastMeasureTime >= MEASURE_INTERVAL){
305      lastMeasureTime = currentTime;
306      voltage_checker();
307    }
308
309    esp_task_wdt_reset();  // Reset watchdog to prevent reboot
310
```

Εικόνα 110:Κώδικας εντός συνάρτησης loop() 1 - WDT, Voltage checker

Αυτό το τμήμα του κώδικα διαχειρίζεται την επικοινωνία μεταξύ της κύριας συσκευής (ESP32) και μιας δευτερεύουσας μονάδας (ESP32 CAM) μέσω λογισμικού σειριακής θύρας (softSerial). Όταν υπάρχουν διαθέσιμα δεδομένα, το σύστημα διαβάζει την απάντηση και την επεξεργάζεται ως String. Ανάλογα με τον ακέραιο αριθμό εντολής που περιέχεται στην απάντηση (command), εκτελούνται διαφορετικές ενέργειες (βλπ. Εικόνα 111):

- **Εντολές 2 έως 5**: Εμφανίζεται μήνυμα επιβεβαίωσης στην οθόνη ότι τα δεδομένα στάλθηκαν επιτυχώς.
- **Εντολή 6**: Η απάντηση αναλύεται για να εξαχθεί ένα κείμενο που προορίζεται για μετάφραση από αγγλικά σε ελληνικά και αναπαραγωγή μέσω μετατροπής κειμένου σε ομιλία (Google TTS).

- **Εντολή 7**: Εμφανίζεται μήνυμα στην οθόνη ότι η φωτογραφία έχει ανέβει επιτυχώς σε αποθηκευτικό χώρο Azure, και αναπαράγεται σχετικό φωνητικό μήνυμα.

Η χρήση της esp_task_wdt_reset() σε προηγούμενα σημεία του βρόχου εξασφαλίζει την ομαλή εκτέλεση και σε αυτό το στάδιο, αποτρέποντας επανεκκινήσεις κατά τη διάρκεια της επικοινωνίας.

```
302   if (softSerial.available()){
303     Serial.println("softSerial ok");
304     String response = softSerial.readStringUntil('\n'); // Read response and   Print the received strin
305     Serial.println("Received from ESP32 CAM (Slave): " + response);
306     // Convert String to const char* using c_str()
307     int command = atoi(response.c_str());
308     Serial.println("SOFT int:" + String(command));
309 
310     if (command == 2  || command == 3 || command == 4 || command == 5 ){
311       tft.setTextColor(TFT_GREEN);
312       tft.setTextSize(1);
313       tft.setTextDatum(MC_DATUM);
314       tft.drawString("Data sent successfully (" + String(command) +")", 120 , 309);
315       delay(750);
316       tft.setTextColor(TFT_BLACK);
317       tft.drawString("Data sent successfully (" + String(command) +")", 120 , 309);
318     }
319 
320     if (command == 6  ){
321       temp = parseString_from_softSerial(response);
322       temp= translateText(temp);
323 
324       // Generate and play speech
325       generateSpeech_With_GoogleTTL(temp);
326     }
327 
328     if (command == 7  ){
329       if (response.charAt(0) == '7'){
330       response = response.substring(2);  // Remove the first character (7)
331       }
332     tft.setTextColor(TFT_WHITE);    // Set the text color to white
333     tft.setTextSize(1);             // Set the text size (1 = small, 2 = medium, 3 = large)
334     tft.setTextDatum(MC_DATUM);
335     // Display the string at the coordinates (10, 10)
336     tft.drawString("THE PHOTO UPLOADED TO AZURE ", 120, 280);
337     tft.drawString("CONTAINER(CAMERA-ONLY) SUCCESSFULLY", 120, 290);
338     //audio.connecttospeech("THE PHOTO UPLOADED TO AZURE CONTAINER SUCCESSFULLY", "en"); // Google TTS
339     audio.connecttospeech("Η ΦΩΤΟΓΡΑΦΙΑ ΣΤΑΛΘΗΚΕ ΣΤΟ ΚΟΝΤΕΙΝΕΡ  ΜΕ ΕΠΙΤΥΧΙΑ","el-GR"); // Google TTS
340     //audio.connecttospeech(response.c_str(), "en"); // Google TTS
341 
342     }
343   }
```

Εικόνα 111:Κώδικας εντός συνάρτησης loop() 2 - Command 2,3,4,5,6,7

Το παραπάνω απόσπασμα κώδικα διαβάζει τις συντεταγμένες αφής από μια οθόνη TFT και ρυθμίζει δυναμικά την ένταση του ήχου με βάση την αναλογική είσοδο από έναν ελεγκτή έντασης (π.χ. ποτενσιόμετρο). Η τιμή που λαμβάνεται από τον αισθητήρα μετατρέπεται με τη συνάρτηση map() από το εύρος 0–4095 σε 0–20 και χρησιμοποιείται για να οριστεί η ένταση του ήχου μέσω της audio.setVolume(). Τέλος, η audio.loop() καλείται συνεχώς για την ομαλή αναπαραγωγή ήχου (βλπ. Εικόνα 112).

```
350     uint16_t x, y;
351     tft.getTouch(&x, &y);
352   s
353       // Get the volume level
354     volume = map ((analogRead(volControl)), 0, 4095, 0 , 20);
355     // Set the volume
356     audio.setVolume(volume);
357     //Serial.println(volume);
358     // Run audio player
359     audio.loop();
```

Εικόνα 112:Κώδικας εντός συνάρτησης loop() 3 - TFT Touch, Audio, Volume

Αυτό το κομμάτι κώδικα ελέγχει αν έχει γίνει αφή στην οθόνη TFT και ελέγχει αν πατήθηκε το κουμπί απενεργοποίησης (REDBUTTON_X1) για διάφορες λειτουργίες, ανάλογα με την τιμή της μεταβλητής Flag. Ανάλογα με το ποια λειτουργία είναι ενεργή (π.χ. Internet Radio, Camera Only, Azure Voice Recognition), ο κώδικας απενεργοποιεί τη συγκεκριμένη λειτουργία, στέλνει σήμα απενεργοποίησης μέσω softSerial, ανανεώνει την οπτική ένδειξη στην οθόνη καλώντας τη redBtn() και τελικά επανεκκινεί το ESP με ESP.restart(). Ο έλεγχος γίνεται μόνο όταν η μεταβλητή SWITCH_ON_OFF είναι ενεργοποιημένη (ίση με 1) και η αφή γίνεται μέσα στα όρια του προκαθορισμένου κουμπιού (βλπ. Εικόνα 113).

```
362   if (tft.getTouch(&x, &y)){
363     if (SWITCH_ON_OFF==1){
364       if ((x > REDBUTTON_X1) && (x < (REDBUTTON_X1 + REDBUTTON_W1)) && (Flag==1)){
365         if ((y > REDBUTTON_Y1) && (y <= (REDBUTTON_Y1 + REDBUTTON_H1))){
366           Flag=0;
367           Serial.println("INTERNET RADIO OFF AND Radio_FLAG=" + String(Flag));
368           redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,1);
369           softSerial.println("5");
370           delay(500);
371           ESP.restart();
372         }
373       }
374
375       if ((x > REDBUTTON_X1) && (x < (REDBUTTON_X1 + REDBUTTON_W1)) && (Flag==2)){
376         if ((y > REDBUTTON_Y1) && (y <= (REDBUTTON_Y1 + REDBUTTON_H1))){
377           Flag=0;
378           Serial.println("CAMERA ONLY OFF AND Cam_FLAG="+String(Flag));
379           redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,5);
380           softSerial.println("5");
381           delay(1500);
382           ESP.restart();
383         }
384       }
385
386       if ((x > REDBUTTON_X1) && (x < (REDBUTTON_X1 + REDBUTTON_W1)) && (Flag==3)){
387         if ((y > REDBUTTON_Y1) && (y <= (REDBUTTON_Y1 + REDBUTTON_H1))){
388           Flag=0;
389           Serial.println("AZURE VOICE REGOCNITION OFF AND Voice_FLAG="+String(Flag));
390           redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,6);
391           Serial.println("Voice_FLAG="+String(Flag));
392           softSerial.println("5");
393           delay(1500);
394           ESP.restart();
395         }
396       }
397
398       if ((x > REDBUTTON_X1) && (x < (REDBUTTON_X1 + REDBUTTON_W1)) && (Flag==4)){
399         if ((y > REDBUTTON_Y1) && (y <= (REDBUTTON_Y1 + REDBUTTON_H1))){
400           Flag=0;
401           redBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,7);
402           Serial.println("Voice_FLAG="+String(Flag));
403           softSerial.println("5");
404           delay(1500);
405           ESP.restart();
406         }
407       }
408     }
```

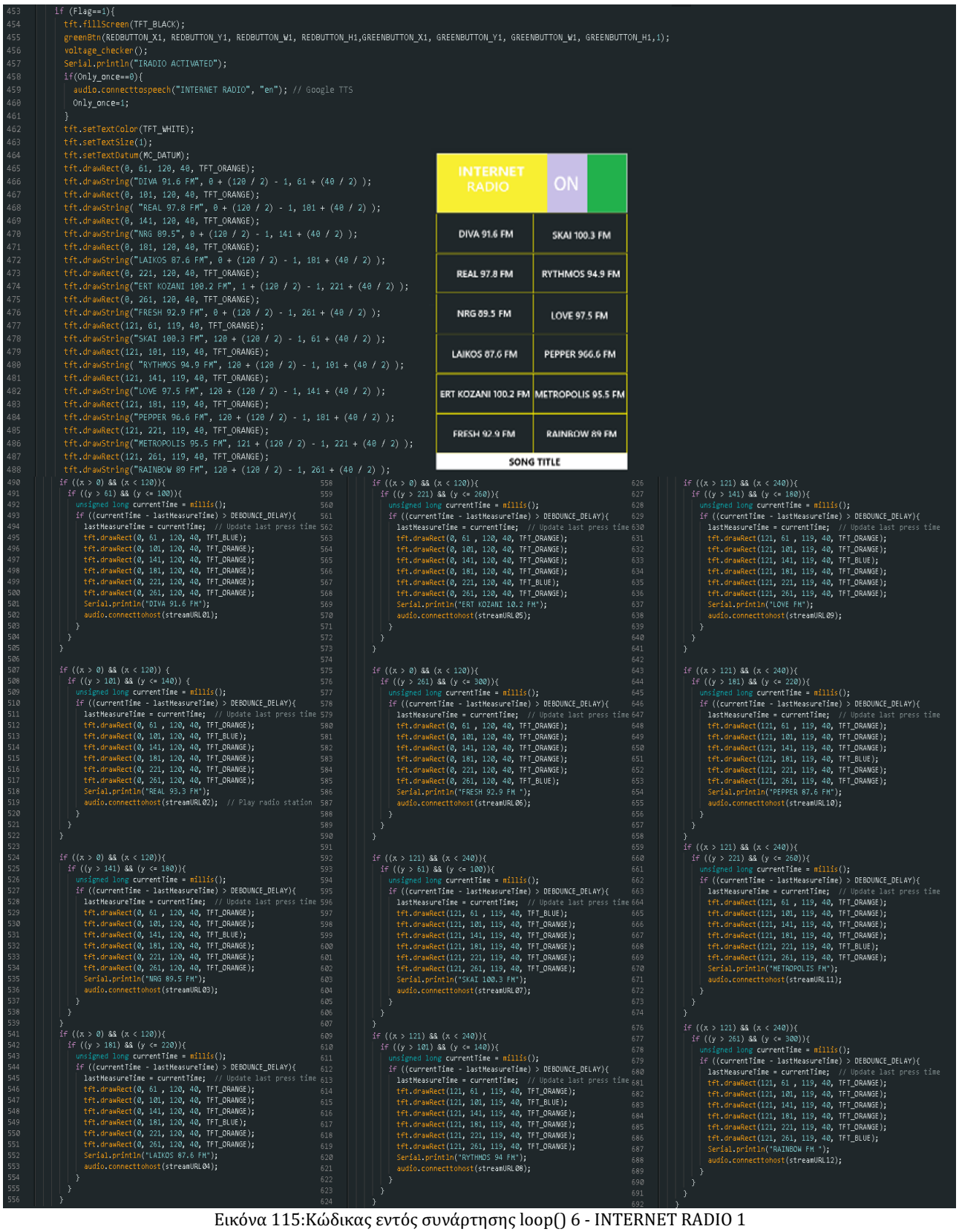
Εικόνα 113:Κώδικας εντός συνάρτησης loop() 4 - Touch off

Αυτό το τμήμα κώδικα ενεργοποιεί μία από τις τέσσερις λειτουργίες του συστήματος όταν το κουμπί ενεργοποίησης (πράσινο) πατηθεί και το σύστημα είναι απενεργοποιημένο (SWITCH_ON_OFF == 0). Για κάθε πράσινο κουμπί, ελέγχεται αν η αφή έγινε μέσα στα όριά του και αν έχει περάσει ο απαραίτητος χρόνος αποφυγής θορύβου (debounce) από την τελευταία ενεργοποίηση. Αν οι συνθήκες πληρούνται, αλλάζει η κατάσταση του SWITCH_ON_OFF σε 1, τίθεται η κατάλληλη τιμή στη μεταβλητή Flag (για Internet Radio, Camera Only, Azure Voice Recognition ή Azure Computer Vision), εμφανίζεται η αντίστοιχη πράσινη ένδειξη στην οθόνη μέσω της greenBtn() και καταγράφεται η ώρα της τελευταίας ενεργοποίησης. Με αυτόν τον τρόπο επιτυγχάνεται διαχείριση πολλαπλών λειτουργιών μέσω αφής με μηχανισμό anti-bounce (βλπ. Εικόνα 114).

```
411     if (SWITCH_ON_OFF == 0){ // RED SQUARE+OFF (SWITCH_ON_OFF == false)
412       if ((x > GREENBUTTON_X1) && (x < (GREENBUTTON_X1 + GREENBUTTON_W1)) && (Flag == 0)){
413         if ((y > GREENBUTTON_Y1) && (y <= (GREENBUTTON_Y1 + GREENBUTTON_H1))){
414           unsigned long currentTime = millis();
415           if ((currentTime - lastMeasureTime) > DEBOUNCE_DELAY){
416             SWITCH_ON_OFF = 1;
417             Flag = 1;
418             Serial.println("INTERNET RADIO ON AND Radio_FLAG=" + String(Flag));
419             greenBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1, GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1, 1);
420             lastMeasureTime = currentTime;
421           }
422         }
423       }
424
425       if ((x > GREENBUTTON_X2) && (x < (GREENBUTTON_X2 + GREENBUTTON_W2)) && (Flag == 0)){
426         if ((y > GREENBUTTON_Y2) && (y <= (GREENBUTTON_Y2 + GREENBUTTON_H2))){
427           unsigned long currentTime = millis();
428           if ((currentTime - lastMeasureTime) > DEBOUNCE_DELAY){
429             SWITCH_ON_OFF = 1;
430             Flag = 2;
431             Serial.println("CAMERA ONLY ON AND Cam_FLAG=" + String(Flag));
432             greenBtn(REDBUTTON_X2, REDBUTTON_Y2, REDBUTTON_W2, REDBUTTON_H2, GREENBUTTON_X2, GREENBUTTON_Y2, GREENBUTTON_W2, GREENBUTTON_H2, 2);
433             lastMeasureTime = currentTime;
434           }
435         }
436       }
437
438       if ((x > GREENBUTTON_X3) && (x < (GREENBUTTON_X3 + GREENBUTTON_W3)) && (Flag == 0)){
439         if ((y > GREENBUTTON_Y3) && (y <= (GREENBUTTON_Y3 + GREENBUTTON_H3))){
440           unsigned long currentTime = millis();
441           if ((currentTime - lastMeasureTime) > DEBOUNCE_DELAY){
442             SWITCH_ON_OFF = 1;
443             Flag = 3;
444             Serial.println("AZURE VOICE RECOGNITION ON AND Voice_FLAG=" + String(Flag));
445             greenBtn(REDBUTTON_X3, REDBUTTON_Y3, REDBUTTON_W3, REDBUTTON_H3, GREENBUTTON_X3, GREENBUTTON_Y3, GREENBUTTON_W3, GREENBUTTON_H3, 3);
446             lastMeasureTime = currentTime;
447           }
448         }
449       }
450
451       if ((x > GREENBUTTON_X4) && (x < (GREENBUTTON_X4 + GREENBUTTON_W4)) && (Flag == 0)){
452         if ((y > GREENBUTTON_Y4) && (y <= (GREENBUTTON_Y4 + GREENBUTTON_H4))){
453           unsigned long currentTime = millis();
454           if ((currentTime - lastMeasureTime) > DEBOUNCE_DELAY){
455             SWITCH_ON_OFF = 1;
456             Flag = 4;
457             Serial.println("AZURE COMPUTER VISION ON AND Photo_FLAG=" + String(Flag));
458             greenBtn(REDBUTTON_X4, REDBUTTON_Y4, REDBUTTON_W4, REDBUTTON_H4, GREENBUTTON_X4, GREENBUTTON_Y4, GREENBUTTON_W4, GREENBUTTON_H4, 4);
459             lastMeasureTime = currentTime;
460           }
461         }
462       }
463     }
```

Εικόνα 114:Κώδικας εντός συνάρτησης loop() 5 - Touch on

Ο κώδικας αυτός υλοποιεί ένα γραφικό περιβάλλον για οθόνη αφής, μέσω του οποίου ο χρήστης μπορεί να επιλέξει και να ακούσει έναν από δώδεκα διαδικτυακούς ραδιοφωνικούς σταθμούς. Όταν ενεργοποιείται, καθαρίζει την οθόνη, εμφανίζει κουμπιά με τα ονόματα των σταθμών, παίζει φωνητικό μήνυμα "INTERNET RADIO" μέσω Google TTS και περιμένει την αφή του χρήστη. Ανάλογα με το πού αγγίζει ο χρήστης, ο κώδικας επιλέγει τον αντίστοιχο σταθμό, τονίζει οπτικά την επιλογή με μπλε περίγραμμα και ξεκινά τη ροή του σταθμού μέσω του αντίστοιχου URL. Περιλαμβάνεται επίσης μηχανισμός αποφυγής πολλαπλών ενεργοποιήσεων λόγω συνεχόμενης αφής, βασισμένος στον χρόνο millis() (βλπ. Εικόνα 115).



Εικόνα 115:Κώδικας εντός συνάρτησης loop() 6 - INTERNET RADIO 1

Αυτό το κομμάτι του κώδικα υλοποιεί τη λειτουργία "CAMERA ONLY" σε μια εφαρμογή με γραφικό περιβάλλον για οθόνη αφής. Όταν η τιμή της μεταβλητής Flag είναι 2, η οθόνη καθαρίζεται και εμφανίζονται οδηγίες ώστε ο χρήστης να πατήσει ένα κουμπί ("ACTIVATE") για να τραβήξει φωτογραφία. Παράλληλα, εκφωνείται το μήνυμα "CAMERA ONLY" μέσω Google TTS μία φορά, και ενεργοποιείται η λειτουργία ελέγχου τάσης. Αν ο χρήστης πατήσει εντός της καθορισμένης περιοχής του κουμπιού, αποστέλλεται το σήμα "2" μέσω σειριακής επικοινωνίας (softSerial) για να δοθεί εντολή ενεργοποίησης της κάμερας (βλπ. Εικόνα 116).

```
696     if (Flag==2){
697       tft.fillScreen(TFT_BLACK);
698       greenBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,5);
699       voltage_checker();
700       Serial.println("CAMERA ONLY ACTIVATED");
701       if(Only_once==0){
702       audio.connecttospeech("CAMERA ONLY", "en"); // Google TTS
703         Only_once=1;
704       }
705       tft.setTextColor(TFT_WHITE);
706       tft.setTextSize(2);
707       tft.setTextDatum(MC_DATUM);
708       tft.drawRect(0, 61, 240, 45, TFT_ORANGE);
709       tft.drawString("PRESS ACTIVATE", 0 + (240 / 2) - 1, 55 + (40 / 2) );
710       tft.drawString("TO TAKE A PHOTO", 0 + (240 / 2) - 1, 72 + (40 / 2) );
711       drawActivateButton(120, 162, 50, TFT_RED, TFT_SILVER, "ACTIVATE");
712       if ((x > 70) && (x < 170)){
713         if ((y > 120) && (y <= 220)){
714           softSerial.println("2");
715           Serial.println("SOFT_DATA=2");
716         }
717       }
718     }
```

Εικόνα 116:Κώδικας εντός συνάρτησης loop() 7 – CAMERA ONLY

Ο παρακάτω κώδικας αφορά τη διαχείριση μιας οθόνης TFT, με σκοπό την ενεργοποίηση φωνητικής αναγνώρισης και την εμφάνιση πληροφοριών στον χρήστη. Όταν η μεταβλητή Flag είναι ίση με 3, το σύστημα καθαρίζει την οθόνη και εμφανίζει κουμπιά για ενεργοποίηση λειτουργιών, ενώ ταυτόχρονα ελέγχει την τάση και ενεργοποιεί την αναγνώριση φωνής μέσω του Google TTS. Αν η μεταβλητή Only_once είναι 0, το σύστημα ενεργοποιεί τη φωνητική αναγνώριση και εκφωνεί τη φράση "VOICE RECOGNITION". Στην οθόνη, εμφανίζεται το μήνυμα ότι η ενότητα είναι για μελλοντική αναβάθμιση και υπάρχει κουμπί για ενεργοποίηση. Αν ο χρήστης αγγίξει την περιοχή της οθόνης που αντιστοιχεί στο κουμπί, το σύστημα εκφωνεί την πληροφορία ότι η ενότητα προορίζεται για αναβάθμιση και στέλνει δεδομένα μέσω σειριακής επικοινωνίας (βλπ. Εικόνα 117).

```
720     if (Flag==3){
721       tft.fillScreen(TFT_BLACK);
722       greenBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,6);
723       voltage_checker();
724       Serial.println("AZURE VOICE REGOCNITION ACTIVATED");
725       if(Only_once==0){
726         audio.connecttospeech("VOICE RECOGNITION", "en"); // Google TTS
727         Only_once=1;
728       }
729       tft.setTextColor(TFT_WHITE);
730       tft.setTextSize(2);
731       tft.setTextDatum(MC_DATUM);
732       tft.drawRect(0, 61, 240, 45, TFT_ORANGE);
733       tft.drawString("THIS SECTION IS", 0 + (240 / 2) - 1, 55 + (40 / 2) );
734       tft.drawString("FOR FUTURE UPGRADE", 0 + (240 / 2) - 1, 72 + (40 / 2) );
735       drawActivateButton(120, 162, 50, TFT_RED, TFT_SILVER, "ACTIVATE");
736       if ((x > 70) && (x < 170)){
737         if ((y > 120) && (y <= 220)){
738           audio.connecttospeech("THIS SECTION IS FOR FUTURE UPGRADE", "en"); // Google TTS
739           softSerial.println("3");
740           Serial.println("SOFT_DATA=3");
741         }
742       }
743     }
```

Εικόνα 117:Κώδικας εντός συνάρτησης loop() 8 – VOICE REGOCNITION

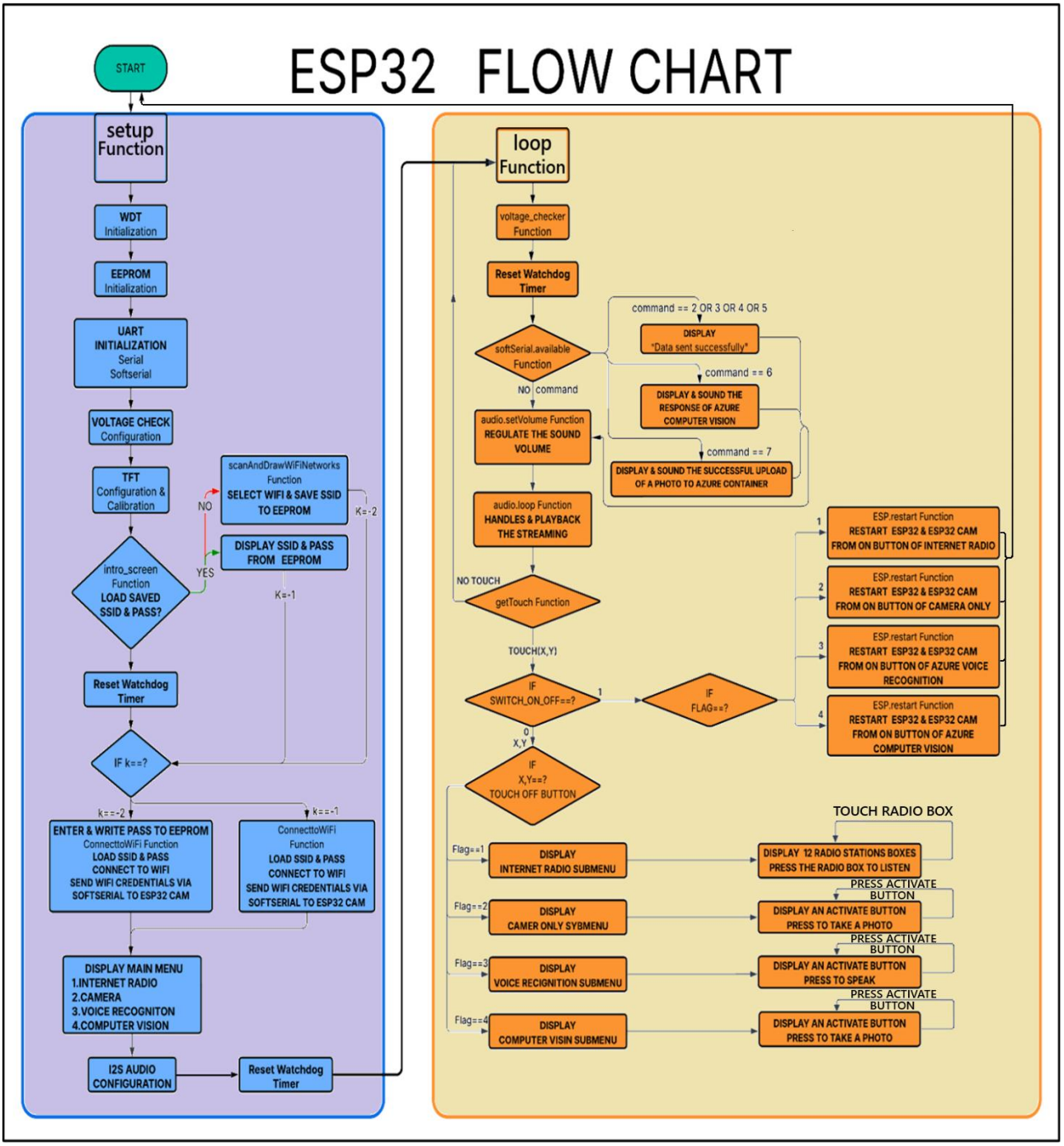
Αυτό το κομμάτι του κώδικα ενεργοποιείται όταν η μεταβλητή Flag είναι ίση με 4 και αφορά τη λειτουργία "Computer Vision". Καθαρίζεται η οθόνη και εμφανίζονται διαδραστικά στοιχεία με οδηγίες προς τον χρήστη να πατήσει το κουμπί "ACTIVATE" για να ξεκινήσει η ανάλυση

εικόνας. Αναπαράγεται μία φορά το ηχητικό μήνυμα "COMPUTER VISION" μέσω Google TTS και αποτυπώνεται μήνυμα επιβεβαίωσης στη σειριακή θύρα. Παρουσιάζονται επιπλέον πεδία στην οθόνη για εμφάνιση χαρακτηριστικών της εικόνας όπως: "CONFIDENCE", "WIDTH" και "HEIGHT", τα οποία θα μπορούσαν να ενημερωθούν δυναμικά από αποτελέσματα υπολογιστικής όρασης. Αν ο χρήστης πατήσει εντός της καθορισμένης περιοχής στην οθόνη αφής, αποστέλλεται ο κωδικός "4" μέσω softSerial, σηματοδοτώντας την ενεργοποίηση της λειτουργίας (βλπ. Εικόνα 118).

```
745     if (Flag==4){
746       tft.fillScreen(TFT_BLACK);
747       greenBtn(REDBUTTON_X1, REDBUTTON_Y1, REDBUTTON_W1, REDBUTTON_H1,GREENBUTTON_X1, GREENBUTTON_Y1, GREENBUTTON_W1, GREENBUTTON_H1,7);
748       voltage_checker();
749       Serial.println("COMPUTERVISION ACTIVATED");
750       if(Only_once==0){
751         audio.connecttospeech("COMPUTER VISION", "en"); // Google TTS
752         Only_once=1;
753         Serial.println("Only_once==1");
754       }
755       tft.setTextColor(TFT_WHITE);
756       tft.setTextSize(2);
757       tft.setTextDatum(MC_DATUM);
758       tft.drawRect(0, 61, 240, 45, TFT_ORANGE);
759       tft.drawString("PRESS ACTIVATE FOR", 0 + (240 / 2) - 1, 53 + (40 / 2) );
760       tft.drawString("COMPUTER VISION", 0 + (240 / 2) - 1, 75 + (40 / 2) );
761       drawActivateButton(120, 162, 50, TFT_RED, TFT_SILVER, "ACTIVATE");
762       tft.setTextColor(TFT_SILVER);
763       tft.fillRect(0, 225, 240, 20, TFT_PURPLE);
764       tft.drawRect(0, 225, 240, 95, TFT_PURPLE);
765       tft.drawRect(0, 280, 240, 20, TFT_PURPLE);
766       tft.drawString("IMAGE DESCRIPTION", 120 , 236);
767       tft.setTextSize(1);
768       tft.setTextDatum(ML_DATUM);
769       tft.drawString("CONFIDENCE:",5,291);
770       tft.drawString("WIDTH:",102,291);
771       tft.drawString("HEIGHT:",169,291);
772       if ((x > 70) && (x < 170)){
773         if ((y > 120) && (y <= 220)){
774           softSerial.println("4");
775           Serial.println("SOFT_DATA=4");
776         }
777       }
778     }
779   }
780 }
```

Εικόνα 118:Κώδικας εντός συνάρτησης loop() 9 – COMPUTER VISION

Για την πληρέστερη κατανόηση της λειτουργίας των δύο βασικών συναρτήσεων, setup() και loop(), παρατίθεται ένα διάγραμμα ροής. Το διάγραμμα αυτό αποτελεί ένα γραφικό εργαλείο απεικόνισης της ακολουθίας εκτέλεσης ενός αλγορίθμου βήμα προς βήμα, διευκολύνοντας την οπτική κατανόηση της ροής (βλπ. Σχήμα 20).



Σχήμα 20:Διάγραμμα ροής συναρτήσεων setup() και loop() (ESP32)

4.3 Σύνοψη τέταρτου κεφαλαίου

Το τέταρτο κεφάλαιο της εργασίας επικεντρώνεται στην αναλυτική παρουσίαση και ερμηνεία των αλγορίθμων που χρησιμοποιήθηκαν για τον προγραμματισμό και τη λειτουργία των συσκευών ESP32 CAM και ESP32. Αρχικά, παρουσιάζονται οι απαιτούμενες βιβλιοθήκες, οι δηλώσεις μεταβλητών καθώς και ο καθορισμός μακροεντολών για κάθε μικροελεγκτή. Στη συνέχεια, αναλύονται οι βοηθητικές συναρτήσεις που εξυπηρετούν επιμέρους λειτουργίες του συστήματος, όπως η επικοινωνία, η λήψη δεδομένων και η ενεργοποίηση εξαρτημάτων. Τέλος, εξετάζεται διεξοδικά η δομή των βασικών συναρτήσεων setup() και loop(), μέσω των οποίων καθορίζεται η αρχικοποίηση και η κυκλική εκτέλεση των λειτουργιών. Το κεφάλαιο είναι χωρισμένο σε δύο ενότητες: μία για το ESP32 CAM και μία για το ESP32, ώστε να γίνει σαφής διαχωρισμός των ρόλων και της λειτουργικότητας κάθε πλακέτας στον συνολικό σχεδιασμό του έργου.

Κεφάλαιο 5:Πειραματική Ανάλυση-Συμπεράσματα-Βελτιώσεις

Το παρόν κεφάλαιο συγκεντρώνει και παρουσιάζει τα αποτελέσματα που προέκυψαν από τη λειτουργία και αξιολόγηση του συστήματος που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας. Μέσα από μια σειρά πειραμάτων, καταγράφονται ποσοτικά και ποιοτικά δεδομένα που αποτυπώνουν την αποτελεσματικότητα και την απόδοση της συσκευής.

Η αξιολόγηση βασίστηκε σε επιλεγμένες μετρικές, κατάλληλες για το αντικείμενο μελέτης, ώστε να εξασφαλιστεί η αντικειμενικότητα και η συγκρισιμότητα των αποτελεσμάτων. Τα πειραματικά σενάρια σχεδιάστηκαν με στόχο να προσομοιώσουν ρεαλιστικές συνθήκες χρήσης και να αποκαλύψουν δυνατά σημεία, αλλά και ενδεχόμενες αδυναμίες του συστήματος.

Στη συνέχεια, γίνεται ανασκόπηση των βασικών συμπερασμάτων που προκύπτουν από την ανάλυση των αποτελεσμάτων, ενώ παράλληλα προτείνονται κατευθύνσεις για μελλοντική βελτίωση και επέκταση της εργασίας. Οι προτάσεις αυτές βασίζονται τόσο στα πειραματικά ευρήματα όσο και στις τεχνολογικές εξελίξεις που δύνανται να ενισχύσουν περαιτέρω την απόδοση ή τη λειτουργικότητα του συστήματος. Στόχος του κεφαλαίου είναι να προσφέρει μια συνολική εικόνα για την επίτευξη των στόχων της εργασίας και να εντοπίσει σημεία προς βελτίωση.

5.1 Αποτελέσματα συστήματος

Η παρούσα ενότητα παρουσιάζει τα αποτελέσματα που προέκυψαν από τη λειτουργική δοκιμή και αξιολόγηση του συστήματος που αναπτύχθηκε στο πλαίσιο της εργασίας. Στόχος είναι η αποτύπωση της πραγματικής απόδοσης του συστήματος υπό συνθήκες που προσομοιώνουν την αναμενόμενη χρήση του. Η ανάλυση βασίζεται σε μετρήσιμα δεδομένα που συλλέχθηκαν μέσω πειραμάτων, τα οποία σχεδιάστηκαν ώστε να καλύψουν κρίσιμες πτυχές της λειτουργικότητας, της αξιοπιστίας και της ενεργειακής συμπεριφοράς του συστήματος. Ιδιαίτερη έμφαση δίνεται στην αξιολόγηση της σταθερότητας, της απόκρισης σε πραγματικό χρόνο και της κατανάλωσης ενέργειας. Τα αποτελέσματα συνοδεύονται από γραφήματα και

τεχνικές παρατηρήσεις, ώστε να αναδειχθούν οι επιδόσεις του συστήματος αλλά και να εντοπιστούν πιθανά περιθώρια βελτίωσης.

Επίσης, η ενότητα αυτή εστιάζει στην ποσοτική και ποιοτική αξιολόγηση της απόδοσης του συστήματος υπολογιστικής όρασης που αναπτύχθηκε. Η αξιολόγηση πραγματοποιήθηκε με βάση μετρήσιμες παραμέτρους, οι οποίες αντανακλούν την αποδοτικότητα, την αξιοπιστία και την πρακτική χρησιμότητα του συστήματος.
Συγκεκριμένα, εξετάζεται:

- **Ο συνολικός χρόνος συνεχούς λειτουργίας** του συστήματος σε πραγματικές ή προσομοιωμένες συνθήκες, γεγονός που συνδέεται άμεσα με την ενεργειακή απόδοση και τη διαχείριση πόρων. Η μεθοδολογία που εφαρμόστηκε για τη μέτρηση του συνολικού χρόνου λειτουργίας του συστήματος βασίστηκε στην παρακολούθηση της κατανάλωσης ενέργειας, από τη στιγμή κατά την οποία η μπαταρία έπαυσε να παρέχει επαρκή ισχύ για τη λειτουργία της συσκευής, έως την πλήρη επαναφόρτισή της. Η διαδικασία αυτή επέτρεψε την ακριβή αποτίμηση του κύκλου εκφόρτισης-φόρτισης, προσφέροντας χρήσιμα δεδομένα για την ενεργειακή συμπεριφορά του συστήματος υπό ρεαλιστικές συνθήκες χρήσης. Συγκεκριμένα, μετρήθηκε η συνολική ποσότητα ενέργειας που αποθηκεύτηκε κατά τη φόρτιση (σε mAh) (βλπ. Εικόνα 119), καθώς και με αμπερόμετρο υπολογίστηκε η κατανάλωση ρεύματος (σε mA) για κάθε διαφορετική περίπτωση λειτουργίας (βλπ. Πίνακα 10). Με βάση τα δύο αυτά δεδομένα, τη χωρητικότητα της μπαταρίας και την κατανάλωση ρεύματος ανά λειτουργία, υπολογίστηκε ο εκτιμώμενος χρόνος συνεχούς λειτουργίας για κάθε επιμέρους σενάριο, σύμφωνα με τον τύπο: **Time = Battery capacity / Amperage** (βλπ. Πίνακα 11).

Εικόνα 119:Ένδειξη χωρητικότητας μπαταρίας

Πίνακας 10: Ρεύματα λειτουργίας συσκευής

Ρεύματα λειτουργίας συσκευής σε mA						
Επαναλήψεις	**1**	**2**	**3**	**4**	**5**	**ΜΟ**
ACTIVATED ONLY	304	314	332	311	321	**316,4**
INTERNET RADIO 25% ΕΝΤΑΣΗ	367	373	398	379	398	**383**
INTERNET RADIO 50% ΕΝΤΑΣΗ	438	444	456	470	469	**455,4**
INTERNET RADIO 75% ΕΝΤΑΣΗ	504	528	539	522	522	**523**
INTERNET RADIO 100% ΕΝΤΑΣΗ	571	570	578	577	570	**573,2**
CAMERA	373	371	390	383	376	**378,6**
COMPUTER VISION	384	389	394	386	393	**389,2**

Πίνακας 11:Χρόνοι λειτουργίας συσκευής

Χρόνοι λειτουργίας συσκευής σε Ώρες:Λεπτά						
Επαναλήψεις	**1**	**2**	**3**	**4**	**5**	**ΜΟ**
ACTIVATED ONLY	2:53	2:48	2:39	2:49	2:44	**2:32**
INTERNET RADIO 25% ΕΝΤΑΣΗ	2:24	2:21	2:12	2:19	2:12	**2:07**
INTERNET RADIO 50% ΕΝΤΑΣΗ	2:00	1:59	1:55	1:52	1:52	**1:48**
INTERNET RADIO 75% ΕΝΤΑΣΗ	1:44	1:40	1:38	1:41	1:41	**1:36**
INTERNET RADIO 100% ΕΝΤΑΣΗ	1:32	1:32	1:31	1:31	1:32	**1:55**
CAMERA ONLY	2:21	2:22	2:15	2:18	2:20	**2:17**
COMPUTER VISION	2:17	2:15	2:14	2:16	2:14	**2:15**

- **Η χρονική απόκριση (latency)** ενός συστήματος αποτελεί κρίσιμο παράγοντα για την αξιολόγηση της αποτελεσματικότητάς του, ιδιαίτερα σε εφαρμογές πραγματικού χρόνου όπως η ανίχνευση αντικειμένων και αποθήκευση αρχείων στο blob container. Η χρονική απόκριση ορίζεται ως το χρονικό διάστημα που μεσολαβεί από τη στιγμή λήψης της εικόνας ή του σήματος μέχρι την παραγωγή της απόφασης ή της εξαγόμενης πληροφορίας.

  Στον παρακάτω πίνακα (βλπ. Πίνακα 12) παρουσιάζονται διαφορετικά σενάρια λειτουργίας του συστήματος και ο αντίστοιχος χρόνος εκτέλεσης, δηλαδή ο χρόνος που απαιτείται για την ολοκλήρωση κάθε διαδικασίας. Οι παράμετροι που αξιολογούνται περιλαμβάνουν την επικοινωνία μέσω WiFi και Internet Radio, την καταγραφή εικόνας με κάμερα και αποστολή της στο blob container (Camera Only), την καταγραφή εικόνας με κάμερα και την εκτέλεση αλγορίθμων υπολογιστικής όρασης (Computer Vision), καθώς και την επανεκκίνηση του συστήματος (Restart). Για κάθε ένα από τα παραπάνω σενάρια, εξετάζεται επίσης η επίδραση της ταυτόχρονης λειτουργίας φωνητικής επεξεργασίας **(**Voice**)**, η οποία ενδέχεται να επηρεάζει τον συνολικό χρόνο εκτέλεσης. Οι χρόνοι εκτέλεσης που παρουσιάζονται έχουν υπολογιστεί με μηχανική μέτρηση (χρονόμετρο) και παρέχουν μια ρεαλιστική εικόνα της απόδοσης του συστήματος σε διαφορετικά περιβάλλοντα και συνθήκες χρήσης.

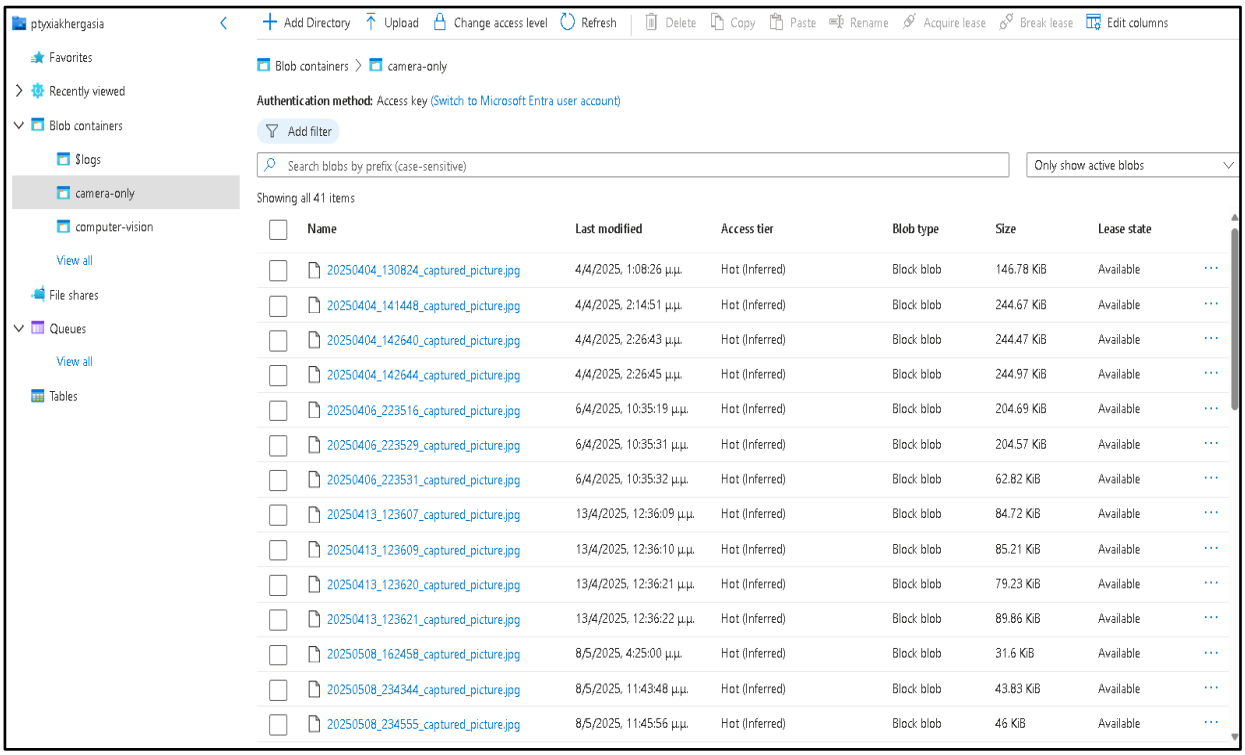
Πίνακας 12:Χρονική απόκριση (Latency)

Χρονική απόκριση (Latency) σε Secs						
Επαναλήψεις	**1**	**2**	**3**	**4**	**5**	**ΜΟ**
Χρόνος εκτέλεσης έναρξης	0,3	0.3	0.3	0.3	0.3	**0,30**
Χρόνος σύνδεσης σε Wifi	3,2	3,4	3,2	3,3	3,1	**3,24**
Χρόνος σύνδεσης σε Wifi+Voice	13,2	13,4	13,2	13,3	13,1	**13,24**
Χρόνος εκτέλεσης Interner radio	0,3	0.3	0.3	0.3	0.3	**0,30**
Χρόνος εκτέλεσης Interner radio+Voice	3,2	3,4	3,5	3,5	3,6	**3,44**
Χρόνος εκτέλεσης Interner radio+Voice +Station	3,5	3,6	3,2	3,3	3,1	**3,34**
Χρόνος εκτέλεσης Camera only	0,3	0.3	0.3	0.3	0.3	**0,30**
Χρόνος εκτέλεσης Camera only+Voice	3,0	2,9	2,9	3,0	3,0	**2,96**
Χρόνος εκτέλεσης Camera only+Voice+Photo	8,0	8,1	8,2	7,9	7,8	**8,00**
Χρόνος εκτέλεσης Computer vision	0,3	0.3	0.3	0.3	0.3	**0,30**
Χρόνος εκτέλεσης Computer vision+Voice	3,0	3,0	3,0	3,0	3,0	**3,00**
Χρόνος εκτέλεσης Computer vision+Voice Response	10,0	11,0	12,0	11,0	11,0	**11,00**
Χρόνος εκτέλεσης Restart	3,2	3,2	3,3	3,3	3,2	**3,24**

Η αξιολόγηση αυτών των χρόνων είναι απαραίτητη για τον σχεδιασμό αποδοτικών και αξιόπιστων συστημάτων, όπου η καθυστέρηση πρέπει να διατηρείται σε ελάχιστα επίπεδα, ώστε να διασφαλίζεται η έγκαιρη και ακριβής απόκριση του συστήματος στις συνθήκες του περιβάλλοντος.

- **Επιλογή Ανάλυσης, Ποιότητας Συμπίεσης και Εκτίμηση Απαιτήσεων Bandwidth**. Για τις ανάγκες της εργασίας, επιλέχθηκε η ανάλυση SXGA (Super Extended Graphics Array), η οποία αντιστοιχεί σε διαστάσεις 1280 x 1024 pixels. Η ανάλυση αυτή προσφέρει υψηλή ευκρίνεια και επαρκή λεπτομέρεια για πλήθος εφαρμογών, ενώ ταυτόχρονα διατηρεί το μέγεθος αρχείου σε αποδεκτά επίπεδα. Κατά την παραμετροποίηση της κάμερας, ως μορφή συμπίεσης εικόνας χρησιμοποιήθηκε το πρότυπο JPEG, με τιμή ποιότητας jpeg_quality = 10. Η συγκεκριμένη παράμετρος κυμαίνεται από 0 έως 63, όπου η τιμή 0 αντιστοιχεί στη μέγιστη δυνατή ποιότητα με τη μικρότερη δυνατή συμπίεση, ενώ η τιμή 63 στην ελάχιστη ποιότητα με τη μεγαλύτερη συμπίεση. Η επιλογή της τιμής 10 αποτελεί έναν ικανοποιητικό συμβιβασμό μεταξύ ποιότητας εικόνας και μεγέθους αρχείου.

  Ενδεικτικά, τιμές υψηλής ποιότητας (στην περιοχή 90–100 της αντίστοιχης κλίμακας 0–100) οδηγούν συνήθως σε μέγεθος αρχείου της τάξης των 300–500 KB για εικόνες υψηλής ανάλυσης. Στην περίπτωση χρήσης ανάλυσης SXGA (1280 × 1024 pixels) και JPEG ποιότητας 10, το μέγεθος του αρχείου κυμαίνεται κατά προσέγγιση μεταξύ 30 και 250 KB, ανάλογα με την περιεκτικότητα της εικόνας σε πληροφορία (πολυπλοκότητα σκηνής, χρωματικές μεταβολές κ.λπ.). Η εκτίμηση αυτή επιβεβαιώνεται και από τα αποτελέσματα που προέκυψαν κατά τη λειτουργία του camera-only blob container, όπου διαπιστώθηκε ότι τα παραγόμενα αρχεία εικόνας εντάσσονται εντός του παραπάνω εύρους μεγέθους (βλπ. Εικόνα 120).



Εικόνα 120:Camera-only blob container

Η παραπάνω ρύθμιση επηρεάζει άμεσα τις απαιτήσεις σε **bandwidth**, ιδιαίτερα σε εφαρμογές όπου η μετάδοση εικόνας γίνεται σε πραγματικό χρόνο, όπως στην αποστολή δεδομένων εικόνας σε Cloud-based platforms. Έστω ότι η μετάδοση αφορά **μια εικόνα ανά δευτερόλεπτο** με μέγεθος εικόνας **250 KByte**, τότε η απαιτούμενη χωρητικότητα μετάδοσης (bandwidth) μπορεί να υπολογιστεί ως εξής:

Bandwidth=1 εικόνα/δευτ.×250 KB=250 KB/s=0,244 MB/s≈0,244x8bit=1,952Mbps
Παρακάτω παρατίθεται ένας πίνακας με το απαιτούμενο bandwidth συνάρτηση του χρόνου ανεβάσματος και του όγκου της εικόνας (βλπ. Πίνακα 13).

Πίνακας 13:Πίνακας Bandwidth

Bandwidth Mbps	0,5 sec	1 sec	1,5 secs	2 secs	2,5 secs	3 secs	3,5 secs	4 secs
Image (50KB)	0,781	0,391	0,260	0,195	0,156	0,130	0,112	0,098
Image (100KB)	1,563	0,781	0,521	0,391	0,313	0,260	0,223	0,195
Image (150KB)	2,344	1,172	0,781	0,586	0,469	0,391	0,335	0,293
Image (200KB)	3,125	1,563	1,042	0,781	0,625	0,521	0,446	0,391
Image (250KB)	3,906	1,953	1,302	0,977	0,781	0,651	0,558	0,488
Bandwidth Mbps	**4,5 secs**	**5secs**	**5,5 secs**	**6 secs**	**6,5 secs**	**7 secs**	**7,5 sec**	**8 sec**
Image (50KB)	0,087	0,078	0,071	0,065	0,060	0,056	0,012	0,011
Image (100KB)	0,174	0,156	0,142	0,130	0,120	0,112	0,023	0,022
Image (150KB)	0,260	0,234	0,213	0,195	0,180	0,167	0,035	0,033
Image (200KB)	0,347	0,313	0,284	0,260	0,240	0,223	0,046	0,043
Image (250KB)	0,434	0,391	0,355	0,326	0,300	0,279	0,058	0,054

Αν η σύνδεση στο διαδίκτυο παρέχεται μέσω κινητού τηλεφώνου (tethering), τότε, σύμφωνα με τα δεδομένα του Πίνακα 13, η αποστολή εικόνας απαιτεί μεν περιορισμένο εύρος ζώνης, ωστόσο υπερβαίνει τις δυνατότητες του βασικού 3G [41] (UMTS – Universal Mobile Telecommunications System), καθιστώντας το ανεπαρκές για γρήγορη μεταφορά. Αντίθετα, οι τεχνολογίες HSPA [41] (High-Speed Packet Access, 3.5G) και άνω θεωρούνται ικανοποιητικές, καθώς προσφέρουν επαρκή ταχύτητα μεταφόρτωσης για την αποστολή εικόνων μεγέθους έως και 250 KB σε εύλογο χρόνο (κάτω από 5 δευτερόλεπτα). Στην πράξη, όμως, για την αξιόπιστη και ασφαλή αποστολή εικόνων απαιτείται τουλάχιστον τεχνολογία HSPA+ (3.75G) [41]. Στην Ελλάδα χρησιμοποιούνται κυρίως δίκτυα 4G και 5G, όπου η ελάχιστη καταγεγραμμένη ταχύτητα αποστολής (upload) ξεκινά από 9,68 Mbps [42], καλύπτοντας επαρκώς τις ανάγκες μεταφοράς δεδομένων μέσω tethering. Ωστόσο, η ταχύτητα μεταφόρτωσης επηρεάζεται άμεσα από την ποιότητα και την ισχύ του σήματος του δικτύου 4G ή 5G. Σε περιοχές με ασθενές σήμα, ακόμη και τα σύγχρονα δίκτυα ενδέχεται να παρουσιάσουν μειωμένες επιδόσεις, γεγονός που μπορεί να επηρεάσει αρνητικά τον χρόνο αποστολής των εικόνων και κατ' επέκταση, τη συνολική απόδοση της συσκευής.

Επιπλέον, όταν η σύνδεση πραγματοποιείται μέσω σταθερού Wi-Fi, τα εμπορικά προγράμματα Internet στην Ελλάδα παρέχουν σημαντικά υψηλότερες ταχύτητες αποστολής. Οι ταχύτητες αυτές ξεκινούν από περίπου 5 Mbps και μπορούν να φτάσουν ή και να ξεπεράσουν τα 18–20 Mbps, ανάλογα με τον πάροχο και τον τύπο της σύνδεσης [43]. Σε τέτοιες συνθήκες, ο χρόνος που απαιτείται για την αποστολή εικόνων μειώνεται αισθητά, καθιστώντας τη διαδικασία ιδιαίτερα αξιόπιστη και αποτελεσματική.

- Η **εμπιστοσύνη (confidence)** ενός αλγορίθμου αναγνώρισης αντικειμένων αναφέρεται στο ποσοστό βεβαιότητας με το οποίο το μοντέλο προβλέπει ότι μια συγκεκριμένη κατηγορία ή ετικέτα είναι σωστή. Στην πλατφόρμα Microsoft Azure, οι υπηρεσίες τεχνητής νοημοσύνης, όπως το Azure Computer Vision, **συνοδεύουν κάθε πρόβλεψη με μία τιμή εμπιστοσύνης**, η οποία εκφράζεται συνήθως ως δεκαδικός αριθμός μεταξύ 0 και 1, ή εναλλακτικά ως ποσοστό (%).

  Στην παρούσα εργασία πραγματοποιήθηκαν πέντε πειράματα αναγνώρισης αντικειμένων, χρησιμοποιώντας ως δείγματα έναν αναπτήρα, ένα στυλό, μία μπαταρία, μία πένσα και ένα CD. Για κάθε αντικείμενο καταγράφηκαν τα βασικά μεταδεδομένα, όπως περιγραφή, ανάλυση (ύψος και πλάτος) και επίπεδο εμπιστοσύνης της αναγνώρισης. Στον Πίνακα 14 παρουσιάζονται ενδεικτικά τα αποτελέσματα της πρόβλεψης, τα οποία περιλαμβάνουν τα εξής μεταδεδομένα:

Πίνακας 14:Αποτελέσματα πειραμάτων υπολογιστικής όρασης

Ανεβασμένη φωτογραφία στο container	Ένδειξη στην οθόνη	Μεταδεδομένα
		**Image description** A battery on a white surface **Confidence** 0,75 **Height** 1280 **Width** 1024
		**Image description** A close-up of a pen **Confidence** 0, 82 **Height** 1280 **Width** 1024
		**Image description** A lighter on a white surface **Confidence** 0,80 **Height** 1280 **Width** 1024
		**Image description** A close-up of a plier **Confidence** 0,79 **Height** 1280 **Width** 1024
		**Image description** A close-up of a cd **Confidence** 0,82 **Height** 1280 **Width** 1024

5.2 Συμπεράσματα

Η διπλωματική εργασία πέτυχε τους στόχους της, τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο. Η ανάλυση, ο σχεδιασμός και η υλοποίηση του συστήματος απέδειξαν τη δυνατότητα εφαρμογής σύγχρονων τεχνολογιών όπως η υπολογιστική όραση σε ρεαλιστικά περιβάλλοντα.

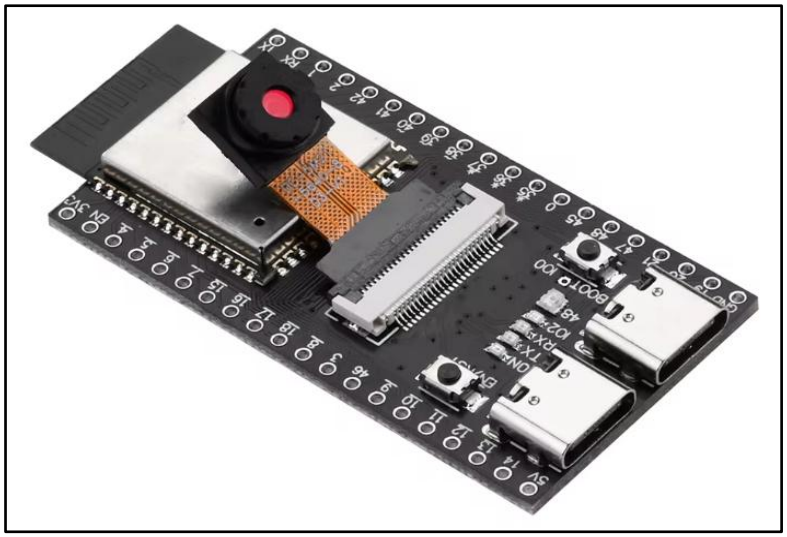
Το ενσωματωμένο σύστημα αναπτύχθηκε και αξιολογήθηκε τόσο σε λειτουργικό όσο και σε υπολογιστικό επίπεδο. Οι δοκιμές έδειξαν ότι η ενσωμάτωση τεχνητής νοημοσύνης λειτούργησε αποτελεσματικά, με ικανοποιητική ακρίβεια και αξιοπιστία ακόμα και σε περιβάλλον περιορισμένων πόρων. Το σύστημα ανταποκρίθηκε σε πραγματικό χρόνο με ικανοποιητικό χρόνο καθυστέρησης και η κατανάλωση ενέργειας παρέμεινε εντός των επιθυμητών ορίων, γεγονός κρίσιμης σημασίας για τα ενσωματωμένα (embedded) συστήματα.

Η εργασία πέτυχε τους βασικούς στόχους της, αποδεικνύοντας ότι είναι εφικτή η ενσωμάτωση τεχνητής νοημοσύνης σε συστήματα χαμηλού υπολογιστικού κόστους, όπως μικροελεγκτές ή πλατφόρμες τύπου Arduino. Ο σχεδιασμός ακολούθησε αρχές αποδοτικής αρχιτεκτονικής, γεγονός που ενίσχυσε τη σταθερότητα και την επεκτασιμότητα του συστήματος. Η τεχνογνωσία που αποκτήθηκε σε επίπεδο hardware και λογισμικού αποτελεί σημαντική βάση για περαιτέρω έρευνα και εφαρμογές.

5.3 Μελλοντικές βελτιώσεις

Για μελλοντικές βελτιώσεις της υλοποίησής μας, προτείνουμε τις ακόλουθες ενέργειες:

1. **Αντικατάσταση των ξεχωριστών μικροελεγκτών ESP32 και ESP32-CAM με την πλατφόρμα ESP32-S3 CAM**. Η απόφαση για την αντικατάσταση βασίστηκε στην ανάγκη απλοποίησης του συστήματος και βελτίωσης της συνολικής απόδοσης. Η χρήση δύο μικροελεγκτών αύξανε την πολυπλοκότητα της επικοινωνίας, τον αριθμό των απαιτούμενων εξαρτημάτων και την κατανάλωση ενέργειας. Αντιθέτως, η ESP32-S3 CAM (βλπ. Εικόνα 121) συνδυάζει σε μία συσκευή επεξεργαστική ισχύ, κάμερα, υποστήριξη και προηγμένες δυνατότητες σύνδεσης, μειώνοντας δραστικά το κόστος, τον χώρο και την πολυπλοκότητα του κυκλώματος. Επιπλέον, ο συνδυασμός USB OTG και η μεγαλύτερη μνήμη, καθιστούν την ESP32-S3 CAM [44] μια πιο αποδοτική και σύγχρονη επιλογή για συστήματα computer vision.



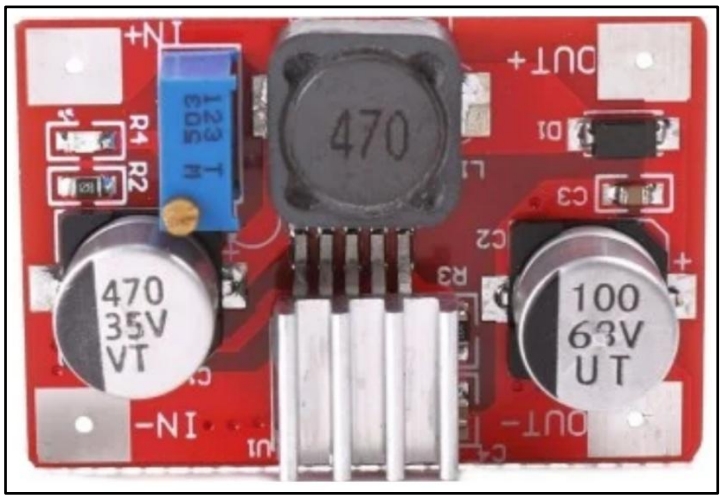
Εικόνα 121:Πλατφόρμα ESP32-S3 CAM

2. **Βελτιστοποίηση Πόρων**: Παρόλο που η παρούσα υλοποίηση επιτυγχάνει ικανοποιητική απόδοση, συνοδεύεται από υψηλές απαιτήσεις σε υπολογιστικούς πόρους. Μελλοντικές βελτιώσεις θα μπορούσαν να στοχεύσουν στη μείωση αυτών των απαιτήσεων, είτε μέσω της αναθεώρησης και βελτίωσης των χρησιμοποιούμενων αλγορίθμων είτε με την υιοθέτηση πιο αποδοτικών τεχνικών σχεδιασμού και υλοποίησης**.**

3. **Αναβάθμιση Συστήματος Τροφοδοσίας**:Μια σημαντική μελλοντική βελτίωση αφορά την αντικατάσταση των υφιστάμενων μπαταριών με μονάδες μεγαλύτερης χωρητικότητας (βλπ. Εικόνα 122). Η αλλαγή αυτή αποσκοπεί στην αύξηση της αυτονομίας του συστήματος, ιδιαίτερα σε περιβάλλοντα όπου η συνεχής επαναφόρτιση ή αντικατάσταση δεν είναι πρακτικά εφικτή. Με τη χρήση μπαταριών υψηλότερης ενεργειακής πυκνότητας, μειώνεται η ανάγκη για συχνή συντήρηση, ενώ διασφαλίζεται η απρόσκοπτη λειτουργία του συστήματος σε απομακρυσμένες ή απαιτητικές συνθήκες λειτουργίας.

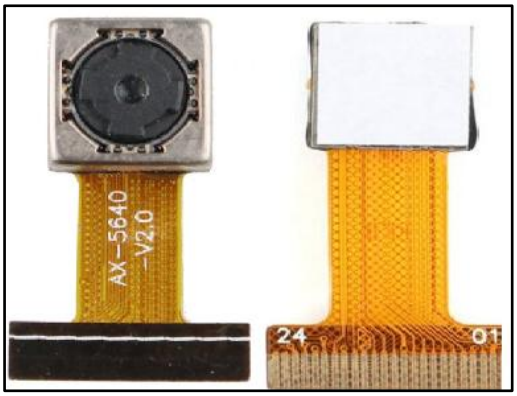
Εικόνα 122:Μπαταρίες Panasonic NCR18650B 3,7V 3200 mah

Στο πλαίσιο της ενεργειακής αναβάθμισης ενσωματώνεται ένα **High Voltage DC-DC Boost Converter** (2A, 5–56V) (βλπ. Εικόνα 123). Η συγκεκριμένη μονάδα επιτρέπει την ανύψωση της τάσης εξόδου από χαμηλά επίπεδα (όπως 3,7 από τις μπαταρίες) σε υψηλότερα επίπεδα ανάλογα με τις ανάγκες (όπως 5V από το esp32). Με αυτόν τον τρόπο, καθίσταται δυνατή η τροφοδοσία εξαρτημάτων που απαιτούν υψηλότερη τάση λειτουργίας, χωρίς την ανάγκη πολλαπλών τροφοδοτικών γραμμών. Επιπλέον, ο μετατροπέας παρέχει σταθερότητα στην τάση εξόδου ακόμη και σε περιπτώσεις όπου η στάθμη φόρτισης της μπαταρίας μεταβάλλεται, ενισχύοντας έτσι τη συνολική αξιοπιστία του συστήματος.



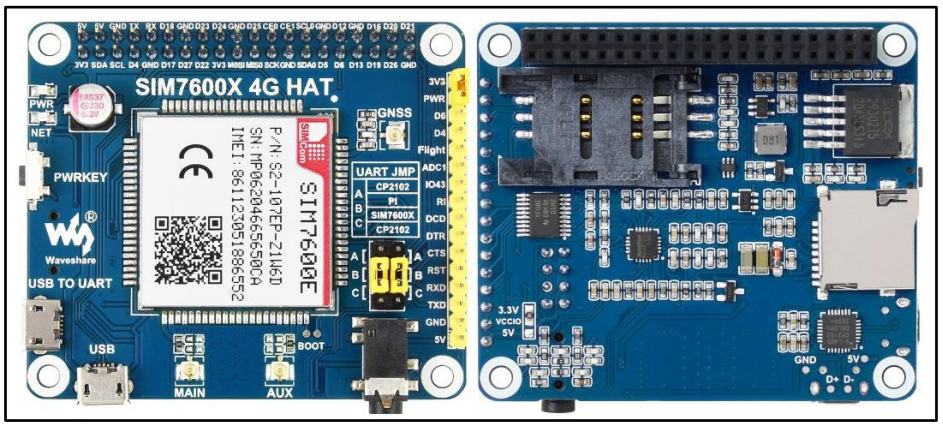
Εικόνα 123:High Voltage DC-DC Boost Converter (2A, 5–56V).

4. **Αναβάθμιση του Αισθητήρα Εικόνας από OV2640 σε OV5640 AF** [45]. Στο πλαίσιο βελτιστοποίησης της ποιότητας εικόνας και της συνολικής απόδοσης του συστήματος όρασης, εξετάστηκε η αντικατάσταση του αισθητήρα εικόνας OV2640 (2MP) με τον πιο εξελιγμένο OV5640 (5MP)(βλπ. Εικόνα 124). Ο OV5640 προσφέρει σημαντικά πλεονεκτήματα, όπως υψηλότερη ανάλυση (έως 2592x1944 pixels), βελτιωμένη ευαισθησία σε συνθήκες χαμηλού φωτισμού και δυνατότητα αυτόματης εστίασης (autofocus), χαρακτηριστικά κρίσιμα για εφαρμογές που απαιτούν ακρίβεια στην αναγνώριση προσώπου ή αντικειμένων. Επιπλέον, υποστηρίζει συμπίεση MJPEG και συμβαδίζει με τις δυνατότητες της πλατφόρμας ESP32-S3 CAM, επιτρέποντας την επεξεργασία εικόνας υψηλότερης ποιότητας χωρίς την ανάγκη εξωτερικής επεξεργασίας.



Εικόνα 124:Αισθητήρας εικόνας OV5640 Auto Focus

5. Η **προσθήκη του SIM7600E-H 4G HAT** [46] στο σύστημα με ESP32 CAM αποτελεί σημαντική αναβάθμιση, καθώς βελτιώνει σημαντικά τη δικτύωση και τη λειτουργικότητα της συσκευής. Με τη δυνατότητα σύνδεσης σε δίκτυα κινητής τηλεφωνίας υψηλής ταχύτητας 4G LTE και το ενσωματωμένο GPS, το σύστημα γίνεται πιο ευέλικτο, αξιόπιστο και κατάλληλο για ένα ευρύ φάσμα εφαρμογών IoT, τόσο σε αστικό περιβάλλον όσο και σε απομακρυσμένες περιοχές. Η ενσωμάτωση του SIM7600E-H (βλπ. Εικόνα 125) ως μονάδα παροχής διαδικτυακής σύνδεσης επιτρέπει τη σταθερή και γρήγορη πρόσβαση στο διαδίκτυο, καθιστώντας το σύστημα αυτόνομο και λειτουργικό ακόμα και σε σημεία όπου δεν υπάρχει διαθέσιμο Wi-Fi ή ενσύρματη σύνδεση.



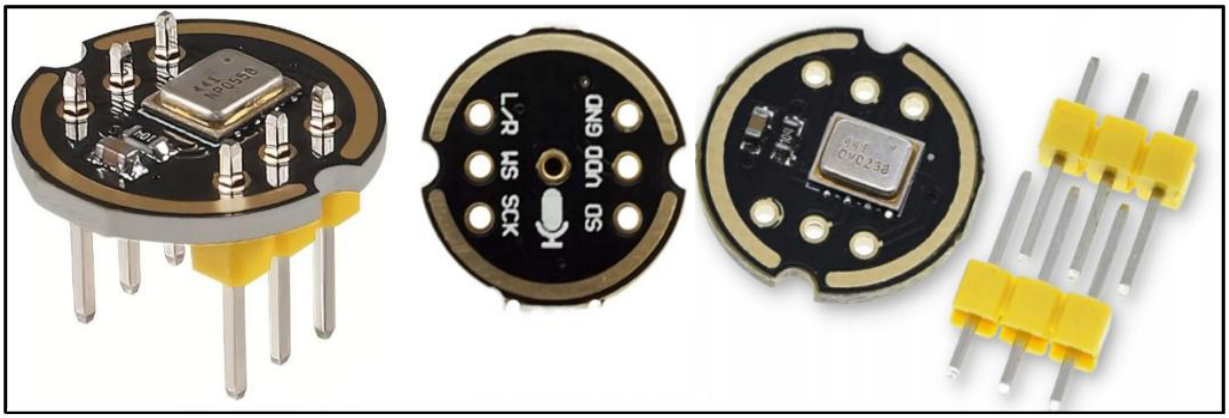
Εικόνα 125:Module SIM7600E-H 4G HAT

6. Προσθήκη του **I²S μικροφώνου INMP441** [47], στο πλαίσιο της παρούσας εργασίας ενσωματώθηκε το ψηφιακό μικρόφωνο INMP441, το οποίο βασίζεται σε τεχνολογία MEMS (Micro-electro-mechanical systems) και διαθέτει ψηφιακή έξοδο τύπου I^2S. Το συγκεκριμένο μικρόφωνο επιλέχθηκε λόγω της υψηλής ποιότητας λήψης ήχου, της εύκολης ενσωμάτωσης με μικροελεγκτές όπως ο ESP32, καθώς και της ενεργειακής του αποδοτικότητας (βλπ. Εικόνα 126).

Το INMP441 επιτρέπει την άμεση ψηφιακή μετάδοση ηχητικών δεδομένων, παρακάμπτοντας την ανάγκη για εξωτερικό μετατροπέα αναλογικού σε ψηφιακό σήμα (ADC). Η σύνδεσή του με τον μικροελεγκτή υλοποιήθηκε μέσω της διεπαφής I^2S, χρησιμοποιώντας τις αντίστοιχες ακίδες WS (Word Select), SCK (Serial Clock) και SD (Serial Data). Για την τροφοδοσία του χρησιμοποιείται τάση 3,3 V, ενώ απαιτείται και κατάλληλη σύνδεση της γείωσης (GND).

Τα ηχητικά δεδομένα που συλλέγονται μπορούν να αποθηκευτούν προσωρινά σε ενδιάμεση μνήμη (buffer RAM), να καταγραφούν σε εξωτερικό αποθηκευτικό μέσο (όπως κάρτα SD) ή να υποβληθούν σε περαιτέρω επεξεργασία σε πραγματικό χρόνο, όπως αναγνώριση φωνής.

Η ενσωμάτωση του INMP441 καθιστά δυνατή την υψηλής ακρίβειας καταγραφή ήχου, προσθέτοντας στο σύστημα σημαντική λειτουργικότητα που μπορεί να αξιοποιηθεί για μελλοντικές επεκτάσεις. Ενδεικτικά, το μικρόφωνο μπορεί να υποστηρίξει λειτουργίες όπως μετατροπή ομιλίας σε κείμενο (speech-to-text), επεκτείνοντας τις δυνατότητες της πλατφόρμας πέρα από την ανάλυση εικόνας (Image Analysis.



Εικόνα 126:Μικρόφωνο I2S INMP441

7. **Αναπαραγωγή Μουσικής από Κάρτα SD**: Η ενσωμάτωση της δυνατότητας αναπαραγωγής μουσικής από κάρτα microSD στοχεύει στην παροχή αυτόνομου μουσικού περιεχομένου. Η συγκεκριμένη λειτουργικότητα θα επιτρέπει στο σύστημα να διαβάζει και να αναπαράγει αρχεία ήχου (π.χ. μορφής WAV ή MP3), τα οποία θα είναι αποθηκευμένα σε κάρτα SD συνδεδεμένη σε κατάλληλη θύρα του μικροελεγκτή (βλπ. Εικόνες 127,128).

Εικόνα 127:Κάρτα SD

Εικόνα 128:Υποδοχή Κάρτα SD

8. **Εισαγωγή περισσότερων ραδιοφωνικών σταθμών,** οι οποίοι θα είναι κατηγοριοποιημένοι με γεωγραφικά κριτήρια, ώστε να ανταποκρίνονται αποτελεσματικότερα στις τοπικές ανάγκες πληροφόρησης και ψυχαγωγίας. Η γεωγραφική κατηγοριοποίηση επιτρέπει τη δημιουργία στοχευμένου περιεχομένου ανά περιοχή, λαμβάνοντας υπόψη τα πολιτισμικά, κοινωνικά και οικονομικά χαρακτηριστικά κάθε τόπου.

   Για παράδειγμα, στην περιοχή της Κοζάνης, όπου κυριαρχεί η ενεργειακή δραστηριότητα, μπορούν να ενισχυθούν σταθμοί που προβάλλουν θέματα περιβάλλοντος, ανάπτυξης και τοπικής οικονομίας. Στη Θεσσαλονίκη, ως μητροπολιτικό κέντρο της Βόρειας Ελλάδας, είναι σημαντική η υποστήριξη σταθμών με έμφαση σε πολιτιστικά, φοιτητικά και πολυπολιτισμικά ζητήματα. Αντίστοιχα, στην Αθήνα, όπου υπάρχει πληθώρα ραδιοφωνικών μέσων, η κατηγοριοποίηση μπορεί να βασιστεί σε γειτονιές ή ειδικά ενδιαφέροντα κοινού (π.χ. ενημέρωση, lifestyle, πολιτική).

   Προσπαθήσαμε επίσης να δημιουργήσουμε τη δυνατότητα μηχανικής εισαγωγής URL ραδιοφωνικού σταθμού από τον χρήστη. Ωστόσο, λόγω του μεγάλου μήκους και της πληθώρας συμβόλων που περιλαμβάνονται συχνά στα URLs, η διαδικασία αυτή κρίθηκε μη φιλική προς τον χρήστη και τελικά απορρίφθηκε, προς όφελος μιας πιο απλοποιημένης και προσβάσιμης εμπειρίας.

9. **Υποστήριξη OTA (Over-the-Air) ενημερώσεων για απομακρυσμένη αναβάθμιση του λογισμικού,** αξιοποιώντας τις δυνατότητες του ESP32. Η τεχνολογία OTA επιτρέπει την απομακρυσμένη αναβάθμιση του firmware, εξαλείφοντας την ανάγκη φυσικής πρόσβασης στη συσκευή για κάθε αλλαγή ή βελτίωση [48]. Το ESP32 παρέχει ενσωματωμένη υποστήριξη OTA μέσω της βιβλιοθήκης ArduinoOTA [49], προσφέροντας σταθερότητα, ασφάλεια και ευκολία στην υλοποίηση τέτοιων μηχανισμών. Με αυτή τη δυνατότητα, κάθε συσκευή μπορεί να ανακτά και να εγκαθιστά νέες εκδόσεις λογισμικού μέσω Wi-Fi, χωρίς να απαιτείται επανεγγραφή μέσω USB ή σειριακής θύρας.

   Η διαδικασία OTA μπορεί να βασιστεί σε αρχιτεκτονική διπλού partitioning (dual app partition), επιτρέποντας στο σύστημα να φορτώνει το νέο firmware στο δεύτερο partition και να εκκινεί από αυτό μόνο εφόσον η εγκατάσταση είναι επιτυχής. Η λήψη των ενημερώσεων μπορεί να πραγματοποιείται μέσω HTTPS, με ταυτόχρονο έλεγχο ψηφιακής υπογραφής ή checksum για την επαλήθευση της ακεραιότητας των δεδομένων. Τα αρχεία .bin μπορούν να φιλοξενούνται είτε σε τοπικό web server, είτε σε cloud πλατφόρμα, προσδίδοντας στο σύστημα ευελιξία και επεκτασιμότητα.
   Η υλοποίηση OTA ενημερώσεων εκτιμάται ότι θα ενισχύσει σημαντικά τη διαχειρισιμότητα και τη μακροχρόνια αξιοπιστία του συστήματος, μειώνοντας τις ανάγκες συντήρησης και επιτρέποντας εύκολες και ασφαλείς αναβαθμίσεις λογισμικού σε πραγματικό χρόνο.

5.4 Σύνοψη πέμπτου κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκαν τα βασικά αποτελέσματα του συστήματος, όπως αυτά προέκυψαν από τη διαδικασία πειραματικής ανάλυσης και αξιολόγησης. Μέσω των δοκιμών, επιβεβαιώθηκε η λειτουργικότητα των κύριων χαρακτηριστικών, ενώ αναδείχθηκαν και σημεία προς περαιτέρω βελτίωση. Τα συμπεράσματα που εξήχθησαν επιβεβαιώνουν τη συμβατότητα της υλοποίησης με τους αρχικούς στόχους, καθώς και τη δυνατότητα κλιμάκωσης και μελλοντικής επέκτασης του συστήματος. Τέλος, προτάθηκαν συγκεκριμένες κατευθύνσεις για μελλοντικές βελτιώσεις, με σκοπό την ενίσχυση της αποδοτικότητας, της ευχρηστίας και της ασφάλειας της πλατφόρμας.

Παραρτήματα

Παράρτημα Α – Κατάλογος και κόστος υλικών

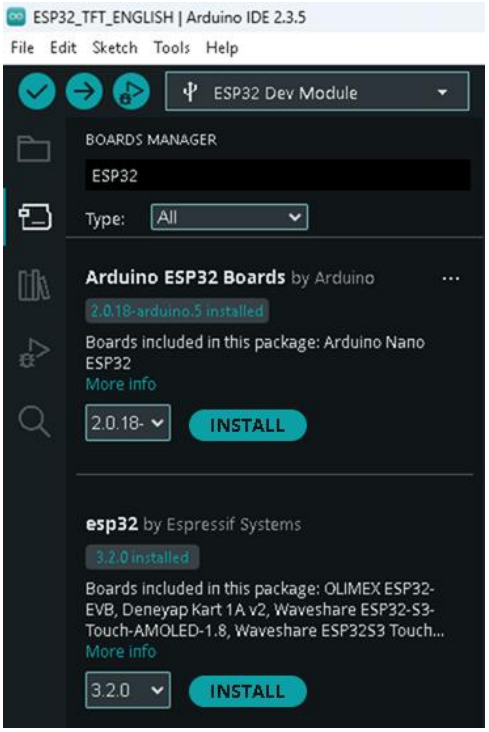
Για την ολοκλήρωση της κατασκευής της συσκευής, χρησιμοποιήθηκαν επιλεγμένα υλικά και εξαρτήματα, τα οποία κρίθηκαν κατάλληλα τόσο ως προς τις απαιτήσεις του συστήματος όσο και ως προς το κόστος. Παρακάτω παρατίθεται αναλυτικός πίνακας με όλα τα υλικά που χρησιμοποιήθηκαν,

Πίνακας 15:Πίνακας και κόστος υλικών

Υλικό	Ποσότητα	Εκτιμώμενο Κόστος Μονάδας σε €
ESP32-WROOM-32	1	5,5
ESP32 cam	1	9,5
ESP32 Cam motherboard	1	1.5
Tft touch ILI9341 3.2 inches	1	8,5
2-Layer PCB –111 mm (Length) × 62 mm (Width) × 2 mm (Thickness)	1	6,5
Μπαταρίες NiMH 1.2V AA	4	6
Θήκη μπαταριών 4xAA	1	0,8
Κουμπί στιγμιαίας χρήσης	3	0,3
Wire 30 AWG	1 μέτρο	0,5
Ηχείο 4Ω 25x36 3W	1	1,2
Adafruit MAX98357A	1	1,85
JST XH συνδέσεις με βήμα 2,54 mm	2	0,3
DC jack 5.5x2.5	1	0,25
Διακόπτης ON/OFF τύπου SPDT	1	0,25
Ποτενσιόμετρο 20K	1	0,35
Voltage checker	1	0,7
3D εκτύπωση με υλικό PLA	105 γραμμάρια	1,05
Σύνολο		**43,55**

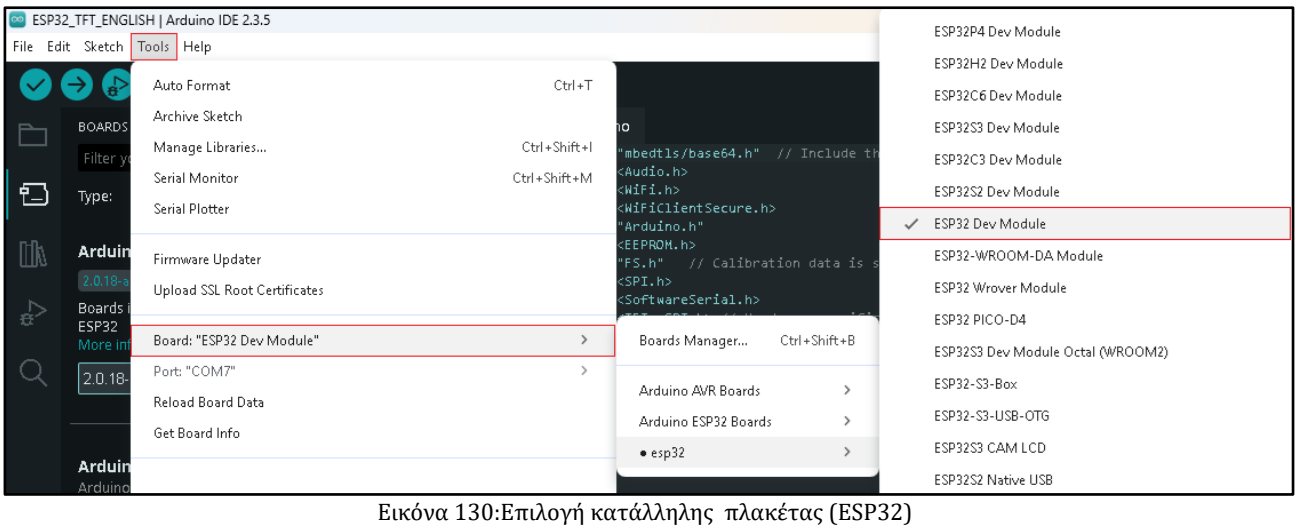
Παράρτημα Β – Παραμετροποίηση Arduino IDE για προγραμματισμό των ESP32 και ESP32 CAM

Μέσα από το περιβάλλον του Arduino IDE, έχουμε τη δυνατότητα να επιλέξουμε τον μικροελεγκτή που επιθυμούμε να προγραμματίσουμε. Στην αριστερή πλευρά του παραθύρου της εφαρμογής υπάρχει μία πλαϊνή μπάρα με διάφορες επιλογές· μία από αυτές είναι ο **Board Manager**. Μέσω αυτής της λειτουργίας μπορούμε να κατεβάσουμε έτοιμα πακέτα υποστήριξης (βιβλιοθήκες και εργαλεία) για διάφορους μικροελεγκτές. Στη γραμμή αναζήτησης πληκτρολογούμε “ESP32” και εγκαθιστούμε το επίσημο πακέτο που έχει αναπτυχθεί από την Espressif για τους μικροελεγκτές ESP32, κάνοντας κλικ στην επιλογή **“Install”** (βλπ. Εικόνα 129).

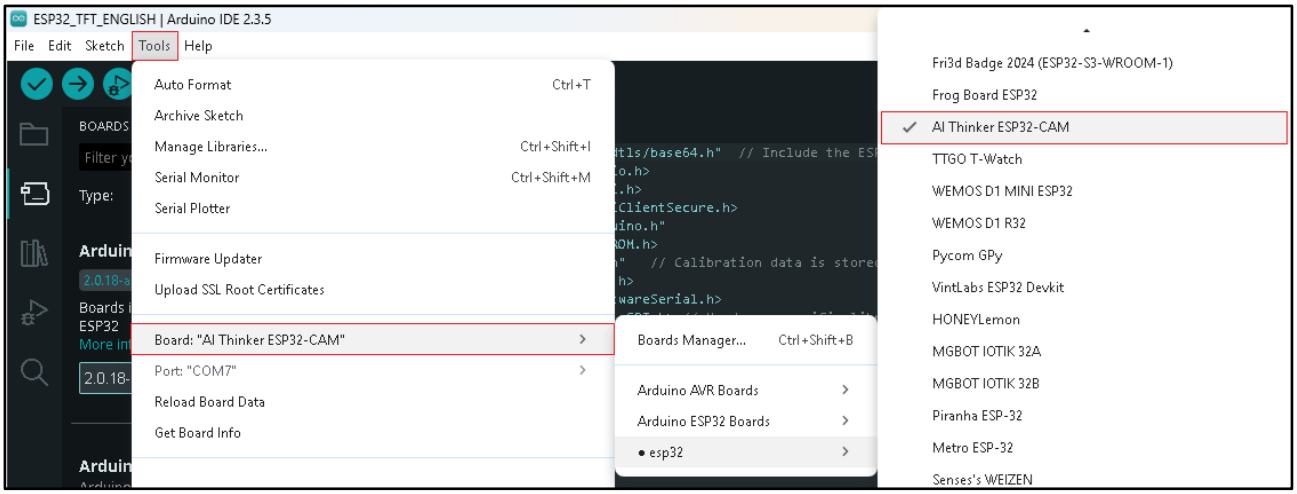


Εικόνα 129:Επιλογή κατάλληλου πακέτου μικροελεγκτών

Αφού ολοκληρωθεί η εγκατάσταση του κατάλληλου πακέτου για τους μικροελεγκτές, μεταβαίνουμε στις ρυθμίσεις και επιλέγουμε την πλακέτα **ESP32** που επιθυμούμε να προγραμματίσουμε. Στην προκειμένη περίπτωση, επιλέγουμε τις πλακέτες **ESP32 Dev Module** και **ESP32-CAM**, ανάλογα με τη συσκευή που χρησιμοποιούμε(βλπ. Εικόνες 130, 131).



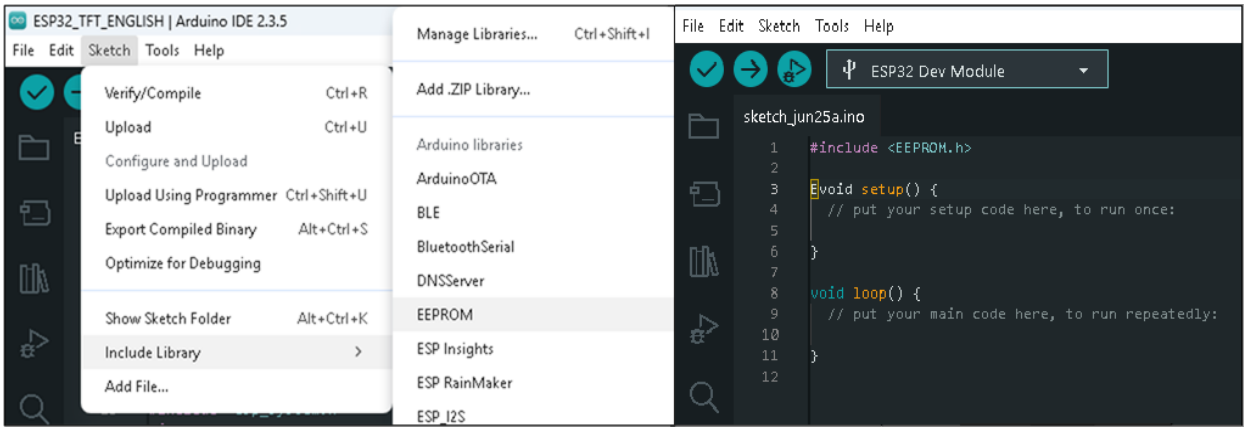
Εικόνα 130:Επιλογή κατάλληλης πλακέτας (ESP32)



Εικόνα 131: Επιλογή κατάλληλης πλακέτας (ESP32 CAM)

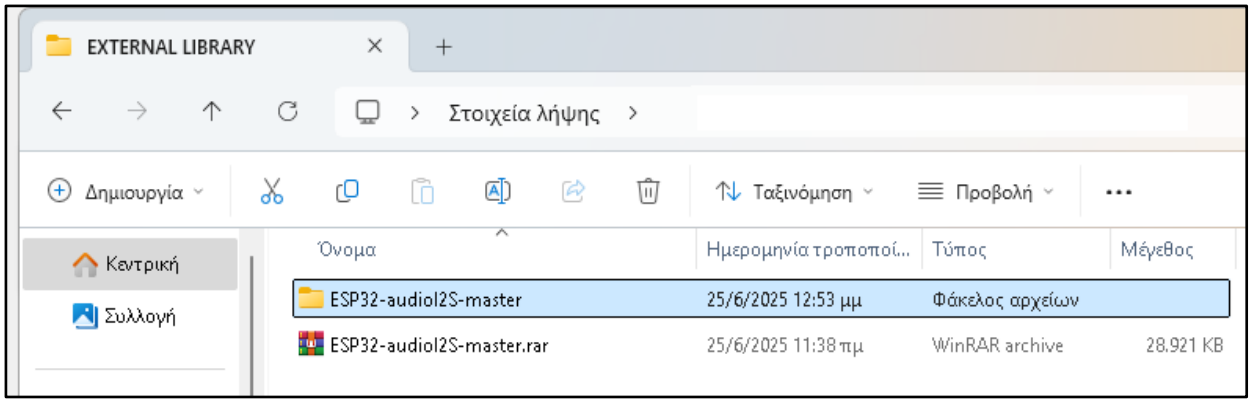
Το επόμενο βήμα, μετά την την επιλογή της πλακέτας, είναι η προσθήκη των απαραίτητων βιβλιοθηκών. Οι βιβλιοθήκες αυτές διακρίνονται σε εσωτερικές (προεγκατεστημένες) και εξωτερικές (προσαρμοσμένες).

- **Εισαγωγή εσωτερικών βιβλιοθηκών**.Το Arduino IDE περιλαμβάνει από προεπιλογή ένα σύνολο εσωτερικών βιβλιοθηκών για βασική λειτουργικότητα, όπως η διαχείριση σειριακής επικοινωνίας, αισθητήρων, εξόδων PWM κ.ά. Για να προσθέσουμε μία τέτοια βιβλιοθήκη στον κώδικά μας, από το μενού, επιλέγουμε: **Sketch > Include Library > [Επιλογή βιβλιοθήκης].** Η εντολή **#include <όνομα_βιβλιοθήκης.h>** προστίθεται αυτόματα στην αρχή του προγράμματος. Οι εσωτερικές βιβλιοθήκες δεν απαιτούν καμία επιπλέον ενέργεια εγκατάστασης(βλπ. Εικόνα 132).

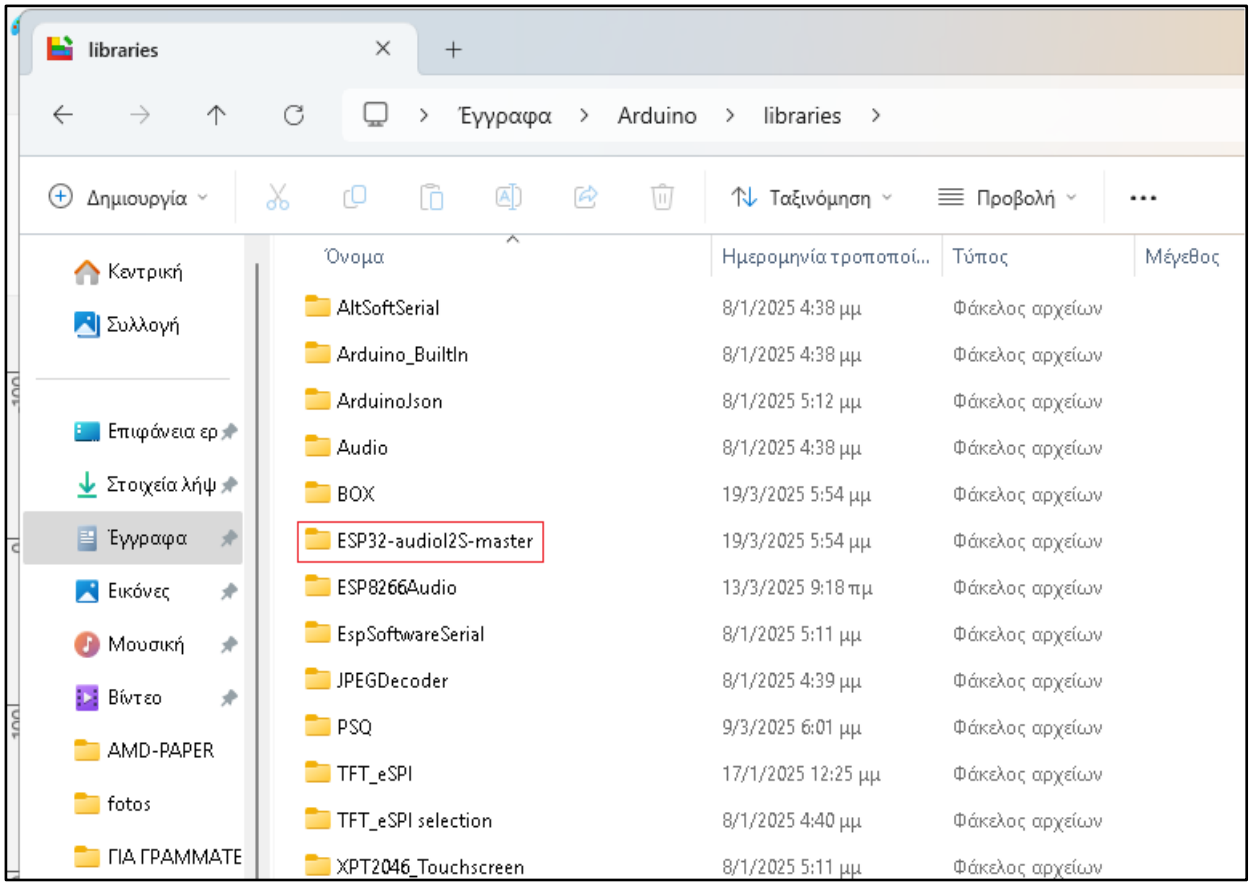


Εικόνα 132:Εισαγωγή εσωτερικών βιβλιοθηκών

- **Χειροκίνητη εγκατάσταση εξωτερικών βιβλιοθηκών**. Η χειροκίνητη εγκατάσταση εξωτερικών βιβλιοθηκών στο Arduino IDE είναι απαραίτητη όταν η βιβλιοθήκη που χρειαζόμαστε δεν είναι διαθέσιμη μέσω του ενσωματωμένου διαχειριστή βιβλιοθηκών. Αρχικά, κατεβάζουμε το αρχείο ZIP της βιβλιοθήκης από την επίσημη πηγή, όπως το GitHub, και στη συνέχεια το αποσυμπιέζουμε στον υπολογιστή μας. Τον φάκελο που περιέχει τα αρχεία της βιβλιοθήκης τον αντιγράφουμε στον κατάλογο **Documents/Arduino/libraries/**, ο οποίος δημιουργείται αυτόματα κατά την εγκατάσταση του Arduino IDE. Αφού ολοκληρωθεί η αντιγραφή, επανεκκινούμε το Arduino IDE ώστε να αναγνωρίσει τη νέα βιβλιοθήκη. Η χρήση της στον κώδικα γίνεται με τον ίδιο τρόπο όπως και με τις εσωτερικές βιβλιοθήκες και η εντολή **#include <όνομα_βιβλιοθήκης.h>** προστίθεται αυτόματα στην αρχή του προγράμματος(βλπ. Εικόνες 133, 134).



Εικόνα 133:Αποσυμπίεση αρχείου ZIP βιβλιοθήκης



Εικόνα 134:Αντιγραφή αρχείου βιβλιοθήκης

Ένας εναλλακτικός τρόπος χειροκίνητης εγκατάσταση εξωτερικών βιβλιοθηκών, περιγράφεται στο Κεφάλαιο 3.5.2 στη σελίδα 70

Παράρτημα Γ – Οδηγίες χρήσης συσκευής

Η παρούσα ενότητα περιγράφει τα βασικά βήματα για την ενεργοποίηση, λειτουργία και επίβλεψη της συσκευής που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας. Οι οδηγίες αποσκοπούν στη διευκόλυνση του τελικού χρήστη, ώστε να αξιοποιήσει πλήρως τις δυνατότητες του συστήματος.

1. **Εκκίνηση συσκευής.** Η συσκευή εκκινεί, θέτοντας τον διακόπτη στη θέση **ON** και ως αποτέλεσμα έχει την εμφάνιση της εισαγωγική οθόνη (βλπ. Εικόνα 135).

2. **Σύνδεση στο Δίκτυο.** Αφού εκκινήσει η συσκευή, εμφανίζεται η εισαγωγική οθόνη, δίνοντας τη δυνατότητα στον χρήστη είτε να φορτώσει τα αποθηκευμένα στοιχεία SSID και PASSWORD πατώντας "**YES**", είτε να επιλέξει και να εισαγάγει μηχανικά νέο SSID και PASSWORD, πατώντας "**NO**" στην οθόνη (βλπ. Εικόνα 135).

   - Με την επιλογή **YES**, φορτώνονται αυτόματα το SSID και το PASSWORD, όπως φαίνεται στην Εικόνα 136 και πραγματοποιείται σύνδεση στο διαδίκτυο (βλπ. Εικόνα 137).

Εικόνα 135:Εκκίνηση συσκευής

Εικόνα 136:Αυτόματη φόρτωση SSID και PASSWORD

Εικόνα 137:Σύνδεση με διαδίκτυο

   - Με την επιλογή **NO**, η συσκευή σκανάρει διαθέσιμα Wi-Fi δίκτυα, όπως φαίνεται στην εικόνα 138 και αφού επιλεχθεί το επιθυμητό SSID(βλπ. Εικόνα 139), στο τέλος εισάγετε το **PASSWORD** μέσω του πληκτρολογίου αφής(βλπ. Εικόνα 140) και τέλος πραγματοποιείται σύνδεση στο διαδίκτυο (βλπ. Εικόνα 141).

Εικόνα 138:Σκανάρισμα διαθέσιμων Wi-Fi δίκτυων

Εικόνα 139:Επιλογή SSID

Εικόνα 140:Εισαγωγή PASSWORD

Εικόνα 141:Σύνδεση με διαδίκτυο

3. **Επιλογή λειτουργίας**. Η συσκευή υποστηρίζει τέσσερις κύριες λειτουργίες: **διαδικτυακό ραδιόφωνο, λειτουργία κάμερας, υπολογιστική όραση και αναγνώριση φωνής**. Η λειτουργία αναγνώρισης φωνής δεν είναι ακόμη ενεργή και προβλέπεται να προστεθεί σε μελλοντική αναβάθμιση.

   - **Επιλογή διαδικτυακού ραδιοφώνου**. Από το μενού της συσκευής, όπως φαίνεται στην Εικόνα 142, επιλέγουμε τη λειτουργία **INTERNET RADIO** πατώντας το γκρι κουμπί **OFF**, το οποίο στη συνέχεια αλλάζει σε **ON**. Αμέσως εμφανίζεται το υπομενού με τους διαθέσιμους ραδιοφωνικούς σταθμούς. Από εκεί μπορούμε να επιλέξουμε τον επιθυμητό σταθμό πατώντας στο αντίστοιχο τετράγωνο κουμπί (βλπ. Εικόνα 143). Κατά την ακρόαση του σταθμού, οποιαδήποτε στιγμή μπορούμε να επιλέξουμε άλλο σταθμό επιλέγοντας άλλο τετράγωνο.

Εικόνα 142:Οθόνη επιλογής λειτουργίας

Εικόνα 143:Οθόνη επιλογής ραδιοφωνικού σταθμού

   - **Επιλογή λειτουργίας φωτογραφικής μηχανής**. Από το μενού της συσκευής, όπως φαίνεται στην Εικόνα 144, επιλέγουμε τη λειτουργία **CAMERA** πατώντας το γκρι κουμπί **OFF**, το οποίο στη συνέχεια αλλάζει σε **ON**. Αμέσως εμφανίζεται το υπομενού (βλπ. Εικόνα 145). Για να τραβήξουμε φωτογραφία, πιέζουμε το κουμπί ACTIVATE.

Εικόνα 144:Οθόνη επιλογής λειτουργίας

Εικόνα 145:Οθόνη λειτουργίας κάμερας

- **Επιλογή λειτουργίας υπολογιστικής όρασης**. Από το μενού της συσκευής, όπως φαίνεται στην Εικόνα 146, επιλέγουμε τη λειτουργία **AZURE COMPUTER VISION** πατώντας το γκρι κουμπί **OFF**, το οποίο στη συνέχεια αλλάζει σε **ON**. Αμέσως εμφανίζεται το υπομενού (βλπ. Εικόνα 147). Για να τραβήξουμε φωτογραφία και να την αποστείλουμε για image description στο AZURE COMPUTER **VISION**, πιέζουμε το κουμπί ACTIVATE. Η επιστρεφόμενη απάντηση απεικονίζεται στην οθόνη (βλπ. Εικόνα 148).

Εικόνα 146:Οθόνη επιλογής λειτουργίας

Εικόνα 147:Οθόνη λειτουργίας κάμερας

Εικόνα 148:Περιγραφή εικόνας

4. **Επανεκκίνηση συσκευής.** Η επανεκκίνηση της συσκευής εκτελείται με το πάτημα του γκρι κουμπιού **ON**, ανεξαρτήτως της ενεργής λειτουργίας. Η διαδικασία οδηγεί στην προβολή της εισαγωγικής οθόνης (βλπ. Εικόνα 135).

Εικόνα 149:Επανεκκίνηση συσκευής

Βιβλιογραφία

[1] Meta Platforms, Inc., "Introducing the New Ray-Ban | Meta Smart Glasses," Meta Newsroom, Sep. 27, 2023. [Online]. Available: https://www.techradar.com/computing/virtual-reality-augmented-reality/ray-ban-meta-smart-glasses-collection-review [Accessed: 17-Mar-2025].

[2] A. Dutta, "Limitless AI Pendant With AI-Powered Audio Recording and Transcription Features Launched; Features, Price," Gadgets 360, Apr. 16, 2024. [Online]. Available: https://www.gadgets360.com/wearables/news/limitless-ai-pendant-price-launch-specifications-features-5453480 [Accessed: 17-Mar-2025].

[3] Humane Inc., "Humane launches Ai Pin," Humane Media Center, Nov. 9, 2023. [Online]. Available: https://www.hp-iq.com/enterprise [Accessed: 17-Mar-2025].

[4] Rabbit Inc., "What is Rabbit R1?", Rabbit Support, 2024. [Online]. Available: https://www.rabbit.tech/support/article/what-is-rabbit-r1 [Accessed: 17-Mar-2025].

[5] Espressif Systems, "ESP32-WROOM-32 Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf [Accessed: 20-Mar-2025].

[6] ESP CAM Module, "DFR0602 ESP32-CAM WiFi Camera Module Datasheet,"[Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf [Accessed: 20-Mar-2025].

[7] ESP CAM Module, "ESP32-CAM WiFi Camera Module Datasheet,"[Online]. Available: https://components101.com/modules/esp32-cam-camera-module [Accessed: 20-Mar-2025].

[8] OmniVision Technologies, Inc., "OV2640 Color CMOS UXGA (2.0 MegaPixel) CAMERACHIP™ with OmniPixel2™ Technology," [Online]. Available: https://www.uctronics.com/download/OV2640_DS.pdf [Accessed: 27-Mar-2025].

[9] R. and M. Ltd, "Embedded Systems Global Market Report 2024 - Research and Markets," Research and Markets, [Online]. Available: Embedded Systems Market Trends, Analysis, Drivers And Forecast To 2033 [Accessed: 10-Apr-2025].

[10] R. S., S. Jahnavi, V. N. Rao, T. T. Ram, and S. S. Ahmed, "Verification of Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) Protocols," IOSR Journal of VLSI and Signal Processing, vol. 14, no. 5, pp. 21–28, Sep.–Oct. 2024. [Online]. Available: https://www.iosrjournals.org/iosr-jvlsi/papers/vol14-issue5/D14052128.pdf [Accessed: 24-Apr-2025].

[11] Circuit Basics, "Basics of UART Communication," Circuit Basics, [Online]. Available: https://www.circuitbasics.com/basics-uart-communication/ [Accessed: 25-Apr-2025].

[12] R. Keim, "Introduction to the I2S Interface," All About Circuits, Mar. 6, 2020. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/introduction-to-the-i2s-interface/ [Accessed: 27- Apr -2025]

[13] Wikipedia contributors, "IEEE 802.11," Wikipedia, The Free Encyclopedia, [Online]. Available: https://en.wikipedia.org/wiki/IEEE_802.11 [Accessed: 29- Apr -2025].

[14] Autodesk Inc., "Autodesk Fusion | 3D CAD, CAM, CAE, & PCB Cloud-Based Software," Autodesk, [Online]. Available: https://www.autodesk.com/eu/products/fusion-360/overview?term=1-YEAR&tab=subscription [Accessed: 7-May-2025]

[15] EasyEDA, "EasyEDA - Online PCB design & circuit simulator," EasyEDA, [Online]. Available: https://easyeda.com/ [Accessed: 7-May-2025]

[16] "EasyEDA," Wikipedia, The Free Encyclopedia, 4 Jun 2025. [Online]. Available: https://en.wikipedia.org/wiki/EasyEDA [Accessed: 7-May-2025]

[17] Shenzhen Creality 3D Technology Co., Ltd., “Creality Print Slicing Software,” Creality, [Online]. Available: https://www.creality.com/pages/creality-print-software. [Accessed: 7-May-2025]

[18] Microsoft Corporation, “Microsoft Azure: Cloud Computing Services,” Microsoft Azure, [Online]. Available: https://azure.microsoft.com/en-us [Accessed: 7-May-2025]

[19] Microsoft, "Azure AI services – Computer Vision documentation," Microsoft Learn, [Online]. Available: https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview [Accessed: 7-May-2025]

[20] Microsoft, "Overview of containers in Azure," Microsoft Learn, [Online]. Available: https://learn.microsoft.com/en-us/azure/storage/blobs/ [Accessed: 9-May-2025]

[21] Arduino AG, “Arduino IDE – Open-source electronics platform,” Arduino, [Online]. Available: Arduino IDE | Arduino Documentation [Accessed: 9-May-2025]

[22] Ilitek, "ILI9341 TFT LCD Single Chip Driver," Ilitek, 2011. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf [Accessed: 10-May-2025]

[23] Adafruit Industries, "Adafruit MAX98357 I2S Class-D Mono Amp," [Online]. Available: https://cdn-shop.adafruit.com/product-files/3006/MAX98357A-MAX98357B.pdf [Accessed: 10-May-2025]

[24] TMicroelectronics, STM32F103x8 Datasheet - ARM-based 32-bit MCU, rev. 19, Sep. 2023. [Online]. https://www.st.com/resource/en/datasheet/stm32f103c8.pdf [Accessed: 9-May-2025]

[25] Arduino, *Arduino Nano (A000005) Datasheet*, 3.3, modified 05/06/2025. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf [Accessed: 9-May-2025]

[26] Raspberry Pi (Trading) Ltd, *Raspberry Pi 4 Model B Datasheet*, Rel. 1.1, Apr. 2024. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf [Accessed: 9-May-2025]

[27] Power Analog Microelectronics (now Diodes Incorporated), “PAM8403: Filterless 3 W Class-D Stereo Audio Amplifier” (datasheet), DSxxxxx Rev. 1, Nov. 2012. Available: https://www.alldatasheet.com/datasheet-pdf/view/246505/PAM/PAM8403.html [Accessed: 9-May-2025]

[28] Texas Instruments, “PCM5100,1,2 Audio Stereo DAC” (PCM5102 datasheet, SLAS764B, Rev. B, Apr. 2012, revised Sept. 2012), 31 pp. Available: https://www.ti.com/lit/ds/symlink/pcm5102.pdf [Accessed: 9-May-2025]

[29] "DOIT ESP32 DevKit V1 Wi-Fi Development Board - Pinout Diagram & Arduino Reference - CIRCUITSTATE Electronics," [Online]. Available: https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/ [Accessed: 10-May-2025]

[30] Adafruit Industries, "Adafruit MAX98357 I2S Class-D Mono Amp," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-max98357-i2s-class-d-mono-amp.pdf [Accessed: 10-May-2025]

[31] Rui Santos, "Uploading Code to the ESP32-CAM MB Micro USB Board," Random Nerd Tutorials, [Online]. Available: https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb/ [Accessed: 10-May-2025]

[32] “3.2 inch SPI Module User Manual.” [Online]. Available: http://www.lcdwiki.com/res/MSP3218/3.2inch_SPI_Module_MSP3218_User_Manual_EN.pdf [Accessed: 10-May-2025]

[33] J.S.T. Mfg. Co., Ltd., XH Connector Datasheet, Jan. 2021. [Online]. Available: https://www.jst.com/wp-content/uploads/2021/01/eXH-new.pdf [Accessed: 12-May-2025].

[34] E-Switch, "500SSP3S2M1QEB SPDT Slide Switch Datasheet," [Online]. Available: https://gr.mouser.com/datasheet/2/140/500SSP3S2M1QEB-3451365.pdf [Accessed: 12-May-2025].

[35] Handson Technology, "6×6×5 mm Tactile Switch – SPST Datasheet," [Online]. Available: https://www.handsontec.com/dataspecs/switches/Tact%20Switch%206x6.pdf [Accessed: 12-May-2025].

[36] Bourns, Inc., PDB18 Series 17 mm Rotary Potentiometer Datasheet, Oct. 2010. [Online]. Available: https://www.bourns.com/docs/Product-Datasheets/PDB18.pdf [Accessed: 12-May-2025].

[37] Same Sky Devices, PJ-102B DC Power Jack Datasheet, Oct. 30, 2019. [Online]. Available: https://www.sameskydevices.com/product/resource/pj-102b.pdf [Accessed: 12-May-2025].

[38] "American Wire Gauge," Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/American_wire_gauge [Accessed: 12-May-2025].

[39] Wikipedia, "Printed circuit board," Wikipedia, The Free Encyclopedia, 29 May 2025. [Online]. Available: https://en.wikipedia.org/wiki/Printed_circuit_board [Accessed: 12-May-2025].

[40] Wolle schreibfaul1, "ESP32-audioI2S: Play audio files on your ESP32 using I2S and external DACs," GitHub, 2020. [Online]. Available: https://github.com/schreibfaul1/ESP32-audioI2S [Accessed: 2-Jun-2025].

[41] P. Rysavy, *Data Capabilities for GSM Evolution to UMTS*, 3G Americas / Rysavy Research, [Online]. Available: https://rysavy.com/wp-content/uploads/2017/08/2002_11_ data_gsm_ to_umts.pdf [Accessed: 17-Jun-2025].

[42] SpeedChecker, *Greece – May 2025 – SpeedChecker Insights*. [Online]. Available: https://www.speedchecker.com/reports/greece [Accessed: 20-May-2025].

[43] nPerf, *GR – Baromètre des connexions Internet fixes – 2024*, Sept. 2025. [Online]. Available: https://media.nperf.com/files/publications/GR/GR-Barometre-Fixed-connections-nPerf-2024_01092025.pdf [Accessed: 18-Jun-2025].

[44] Espressif Systems, ESP32-S3 Series Datasheet, Version 2.0, 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf [Accessed: 16-May-2025].

[45] OmniVision Technologies, OV5640 5-Megapixel Product Specification, Version 2.03, May 2011. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/OV5640_datasheet.pdf [Accessed: 16-May-2025].

[46] Waveshare, SIM7600E-H 4G HAT for Raspberry Pi, [Online]. Available: https://www.waveshare.com/wiki/SIM7600E-H_4G_HAT [Accessed: 20-May-2025].

[47] InvenSense (TDK), "INMP441 Omnidirectional Microphone with Bottom Port and I^2S Digital Output," Datasheet DS-INMP441-00, Rev. 1.1, 05 May 2014. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf [Accessed: 18-Jun-2025].

[48] E. Sanchez and J. Smith, Designing IoT Devices with ESP32: A Practical Guide, 2nd ed., Packt Publishing, 2021. [Accessed: 18-Jun-2025].

[49] ArduinoOTA Library, "OTA Updates with ESP32," Arduino Project Hub, [Online]. Available: https://github.com/espressif/arduino-esp32/tree/master/libraries/ ArduinoOTA [Accessed: 18-Jun-2025].

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

Ορολογία	Ανάλυση
AES	Advanced Encryption Standard
ARM	Advanced RISC Machine
ASCII	American Standard Code for Information Interchange
AWG	American Wire Gauge
ADC	Analog-to-Digital Converter
ABS	Anti-lock Braking System
API	Application Programming Interface
RSA	Asymmetric Encryption
CPU	Central Processing Unit
CAGR	Compound annual growth rate
DC	Direct Current
ECC	Elliptic Curve Cryptography
eMMC	embedded MultiMediaCard
FPGA	Field-Programmable Gate Array
GPIO	General Purpose Input/Output
GHz	Gigahertz
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HDMI	High-Definition Multimedia Interface
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
I2S	Inter-IC Sound (Inter-Integrated Circuit Sound)
JSON	JavaScript Object Notation
LSB	Least Significant Bit
MPU	Micro Processor Unit
MCU	Microcontroller Unit
mAh	Milliamp Hours
MSB	Most Significant Bit
Ni-MH	Nickel-Metal Hydride

Ορολογία	Ανάλυση
OCR	Optical Character Recognition
PSRAM	Pseudo Static RAM
PCM	Pulse Code Modulation
PWM	Pulse Width Modulation
ROM	Read-Only Memory
RTOS	Real-Time Operating System
Risc	Reduced Instruction Set Computer
REST	Representational State Transfer
RSA	Rivest-Shamir-Adleman (Asymmetric Encryption)
RMS	Root Mean Square
SHA	Secure Hash Algorithm
SPI	Serial Peripheral Interface
SSID	Service Set Identifier
SPDT	Single Pole Double Throw
SPST	Single Pole Single Throw
SBC	Single-Board Computer
SDK	Software Development Kit
SSD	Solid State Drive
SRAM	Static Random Access Memory
SMT	Surface-Mount Technology
SoC	System on a Chip
SXGA	Super Extended Graphics Array
THD+N	Total Harmonic Distortion plus Noise
RNG	True Random Number Generator
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WPA	Wi-Fi Protected Access
βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
π.χ.	παραδείγματος χάριν
ΤΝ	Τεχνητής Νοημοσύνης

Απόδοση Ξενόγλωσσων Όρων

Αγγλική Ορολογία	Ελληνική Μετάφραση
Anti-lock Braking System	Σύστημα Αντιμπλοκαρίσματος Φρένων
Access Point	Σημείο Πρόσβασης
Analog-to-Digital Converter	Μετατροπέας Αναλογικού σε Ψηφιακό
Advanced Encryption Standard	Προηγμένο Πρότυπο Κρυπτογράφησης
Artificial Intelligence	Τεχνητή Νοημοσύνη
Application Programming Interface	Διεπαφή Προγραμματισμού Εφαρμογών
Advanced RISC Machine	Προηγμένος Υπολογιστής RISC
American Standard Code for Information Interchange	Αμερικανικό Πρότυπο Κωδικοποίησης Πληροφοριών
American Wire Gauge	Αμερικανική Κλίμακα Διαμέτρου Συρμάτων
Battery Case	Θήκη Μπαταρίας
Baud Rate	Ρυθμός Baud (μετάδοσης δεδομένων)
Bloatware	Λογισμικό επιβάρυνσης (συνήθως προεγκατεστημένο)
Boot	Εκκίνηση
Boot Mode	Λειτουργία Εκκίνησης
Bootloader	Φορτωτής Εκκίνησης
Buffer	Ενδιάμεση Μνήμη
CAGR - Compound Annual Growth Rate	Σύνθετος Ετήσιος Ρυθμός Ανάπτυξης
Capacitive pins	Ακίδες Αφής (Χωρητικές)
Captioning	Αυτόματη Περιγραφή
Clock	Ρολόι (χρονισμός)
Cloud	Υπολογιστικό Νέφος
Cloud Based Dashboards	Πίνακες ελέγχου βασισμένοι σε cloud
Computer Vision	Υπολογιστική Όραση
Computer-Aided Design	Σχεδίαση με Υπολογιστική Βοήθεια
Computer-Aided Engineering	Μηχανική με Υπολογιστική Βοήθεια
Computer-Aided Manufacturing	Κατασκευή με Υπολογιστική Βοήθεια
Container	Κοντέινερ (σε λογισμικό ή υλικό)
Central Processing Unit	Κεντρική Μονάδα Επεξεργασίας
Data	Δεδομένα
Data Bits	Ψηφία Δεδομένων

Αγγλική Ορολογία	Ελληνική Μετάφραση
Data Memory	Μνήμη Δεδομένων
Direct Current	Συνεχές Ρεύμα
debugging	Αποσφαλμάτωση
Device	Συσκευή
Digital-to-Analog Converter	Μετατροπέας Ψηφιακού σε Αναλογικό
Dimmers	Ροοστάτες (ρυθμιστές φωτεινότητας)
Elliptic Curve Cryptography	Κρυπτογραφία Ελλειπτικής Καμπύλης
Embedded	Ενσωματωμένα
eMMC – embedded MultiMediaCard	Ενσωματωμένη Κάρτα Μνήμης
Firmware	Ενσωματωμένο Λογισμικό
Flags	Σημαίες (ενδεικτικά bits σε κώδικα)
Field-Programmable Gate Array	Διάταξη Προγραμματιζόμενων Πυλών Πεδίου
Framework	Πλαίσιο Ανάπτυξης Λογισμικού
Full Duplex	Πλήρης Αμφίδρομη Επικοινωνία
General Purpose Input/Output	Γενικής Χρήσης Είσοδοι/Έξοδοι
GPS	Παγκόσμιο Σύστημα Εντοπισμού
Graphics Processing Unit	Μονάδα Επεξεργασίας Γραφικών
Graphical User Interface	Γραφικό Περιβάλλον Χρήστη
Hands-free	χωρίς χέρια
Hardware	υλικό (υπολογιστή)
Hardware acceleration	Επιτάχυνση με Υλικό
High-Definition Multimedia Interface	Διεπαφή Υψηλής Ανάλυσης Πολυμέσων
Hibernation	Αναστολή Λειτουργίας (ύπνος συσκευής)
High Voltage	Υψηλή Τάση
I2C	Διασύνδεση Συσκευών με 2 Καλώδια
Inter-IC Sound	Διασύνδεση Ήχου μεταξύ Κυκλωμάτων
Institute of Electrical and Electronics Engineers	Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών
Image	Εικόνα
Image Analysis	Ανάλυση Εικόνας
Infill	Πυκνότητα γέμισης (σε 3D εκτύπωση)
Infrastructure Mode	Λειτουργία Υποδομής (Wi-Fi)
Input Ports	Θύρες Εισόδου
Interface	Διεπαφή
Internet	Διαδίκτυο
Internet of Things	Διαδίκτυο των Πραγμάτων
Internet Radio	Διαδικτυακό Ραδιόφωνο
ISO	Διεθνής Οργανισμός Τυποποίησης
JavaScript Object Notation	Σημειογραφία Αντικειμένων της JavaScript
JST	Τύπος συνδέσμου
Jumper	Βραχυκυκλωτής (καλώδιο γεφύρωσης)
Laser projector	Προβολέας λέιζερ
Leading Edge	Ανερχόμενη Ακμή
Logging	Καταγραφή Δραστηριότητας
Low-power adapters	Αντάπτορες χαμηλής κατανάλωσης
Least Significant Bit	Λιγότερο Σημαντικό Bit
Milliampere-hour	Χωρητικότητα Μπαταρίας
Male	Αρσενική Υποδοχή

Αγγλική Ορολογία	Ελληνική Μετάφραση
Master	Κύριος
Master In Slave Out	Κύριος Είσοδος, Υποτελής Έξοδος
Master Out Slave In	Κύριος Έξοδος, Υποτελής Είσοδος
Mbps	Μεγαμπίτ ανά δευτερόλεπτο
Microcontroller Unit	Μονάδα Μικροελεγκτή
Microcontroller	Μικροελεγκτής
MicroSD	Κάρτα Μνήμης ΜικροSD
Module	Μονάδα
Momentary buttons	Πλήκτρα στιγμιαίας επαφής
Micro Processor Unit	Μικροεπεξεργαστής
Most Significant Bit	Πιο Σημαντικό Bit
Nickel-Metal Hydride	Νικελίου-Μεταλλικού Υδριδίου
Non-volatile	Μη Πτητική
Optical Character Recognition	Οπτική Αναγνώριση Χαρακτήρων
Open-source	Ελεύθερης πρόσβασης
Output Ports	Θύρες Εξόδου
Over-the-Air	Ασύρματη Αναβάθμιση (OTA)
Padding	Συμπλήρωση
Parity Bit	Ψηφίο Ισοτιμίας
Password	Κωδικός Πρόσβασης
PCB	Πλακέτα Τυπωμένου Κυκλώματος
Pulse Code Modulation	Διαμόρφωση Κώδικα Παλμού
Peer-to-Peer	Άμεση Σύνδεση (ομότιμη)
Pin headers	Επαφές/υποδοχές ακίδων
Pins	Ακίδες
PLA	Πολυγαλακτικό Οξύ (υλικό 3D εκτύπωσης)
Pricing tier	Επίπεδο Τιμολόγησης
Program Memory	Μνήμη Προγράμματος
PSRAM – Pseudo Static RAM	Ψευδοστατική Μνήμη RAM
Pulse Code Modulation	Κωδικοποίηση Παλμού
PWM – Pulse Width Modulation	Διαμόρφωση Εύρους Παλμού
Real-Time	Πραγματικού Χρόνου
Redundancy	Πλεονασμός
Remapping	Επαναχαρτογράφηση
Repeater	Αναμεταδότης
RESET	Επαναφορά
Representational State Transfer Application Programming Interface	REST API – Διεπαφή Προγραμματισμού με Βάση το REST
Representational State Transfer	Αντιπροσωπευτική Μεταφορά Κατάστασης
RGB565	Μορφή Χρώματος (16-bit RGB)
Reduced Instruction Set Computer	Υπολογιστής Μειωμένων Εντολών
Root Mean Square	Τετραγωνική Μέση Τιμή

Αγγλική Ορολογία	Ελληνική Μετάφραση
Robotics	Ρομποτική
Read-Only Memory	Μνήμη Μόνο για Ανάγνωση
Rivest-Shamir-Adleman	Μέθοδος Ασύμμετρης Κρυπτογράφησης
Real-Time Operating System	Λειτουργικό Σύστημα Πραγματικού Χρόνου
RX	Λήψη (Receive)
Single-Board Computer	Υπολογιστής σε Μοναδική Πλακέτα
Serial Clock	Σειριακό Ρολόι
Software Development Kit	Πακέτο Ανάπτυξης Λογισμικού
Serial Monitor	Παρακολούθηση Σειριακής Θύρας
Serial Port	Σειριακή Θύρα
Secure Hash Algorithm	Αλγόριθμος Ασφαλούς Κατακερματισμού
Shutdown	Τερματισμός Λειτουργίας
Signal Routing	Δρομολόγηση Σήματος
Simulation Program with Integrated Circuit Emphasis	Πρόγραμμα Προσομοίωσης με Έμφαση στα Ολοκληρωμένα Κυκλώματα (SPICE)
Single Pole, Double Throw	Μονοπολικό, Διπλής Ρίψης
Slave	Υποτελής
Slicer	Λογισμικό τεμαχισμού για 3D εκτύπωση
Surface-Mount Technology	Τεχνολογία Επιφανειακής Τοποθέτησης
System on a Chip	Σύστημα σε Τσιπ
Software	Λογισμικό
Single Pole Double Throw	Διακόπτης Μονού Πόλου Διπλής Κατεύθυνσης
Serial Peripheral Interface	Σειριακή Διεπαφή Περιφερειακών
SPIFFS	Σύστημα Αρχείων Flash για ESP32/ESP8266
Single Pole Single Throw	Διακόπτης Μονού Πόλου Μονής Κατεύθυνσης
Static Random Access Memory	Στατική Μνήμη Τυχαίας Προσπέλασης
Solid State Drive	Μονάδα Στερεάς Κατάστασης
Service Set Identifier	Αναγνωριστικό Ασύρματου Δικτύου
Start Bit	Ψηφίο Έναρξης
Stop Bit	Ψηφίο Τερματισμού
Streaming	Ροή δεδομένων (π.χ. πολυμέσων)
Subscription	Συνδρομή
Supports	Στηρίγματα
Text-to-Speech	Μετατροπή Κειμένου σε Ομιλία
TFT	Τεχνολογία Υγρών Κρυστάλλων (Thin Film Transistor)
THD+N – Total Harmonic Distortion + Noise	Ολική Αρμονική Παραμόρφωση και Θόρυβος
Simestamp	Χρονική Σήμανση
Soggle	Διακόπτης Εναλλαγής
Trailing Edge	Κατερχόμενη Ακμή
TX	Μετάδοση (Transmit)
UART	Ασύγχρονος Δέκτης/Πομπός

Αγγλική Ορολογία	Ελληνική Μετάφραση
USB	Καθολικός Σειριακός Δίαυλος
USB-to-Serial	Μετατροπέας USB σε Σειριακή Θύρα
Volatile	Πτητική
Voltage checker	Ελεγκτής Τάσης
WPA	Προστατευόμενη Ασύρματη Πρόσβαση