Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΣΥΣΤΗΜΑ ΑΝΑΓΝΩΡΙΣΗΣ ΛΕΞΕΩΝ ΣΕ ΜΙΚΡΟΕΛΕΓΚΤΗ ΒΑΣΙΣΜΕΝΟ ΣΕ ΣΥΝΕΛΙΚΤΙΚΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΙΟΣ ΚΟΣΥΒΑΣ

Επιβλέπων: Παναγιώτης Τσανάκας,
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

ΣΥΣΤΗΜΑ ΑΝΑΓΝΩΡΙΣΗΣ ΛΕΞΕΩΝ ΣΕ ΜΙΚΡΟΕΛΕΓΚΤΗ ΒΑΣΙΣΜΕΝΟ ΣΕ ΣΥΝΕΛΙΚΤΙΚΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΙΟΣ ΚΟΣΥΒΑΣ

Επιβλέπων: Παναγιώτης Τσανάκας,
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 4η Οκτωβρίου 2021

.....	.....	.....
Π. Τσανάκα Καθηγητής Ε.Μ.Π.	Δ.Σούντρη Καθηγητής Ε.Μ.Π.	Δ.Κουτσούρη Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2021

.....

Δημήτριος Κόσυβας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Περίληψη

Δημιουργήθηκε ένα σύστημα εντοπισμού λέξεων-κλειδιών σε μία ενσωματωμένη συσκευή χρησιμοποιώντας συνελικτικά νευρωνικά δίκτυα. Ο κύριος στόχος αυτής της διπλωματικής εργασίας ήταν ο σχεδιασμός, η εφαρμογή και η αξιολόγηση μοντέλων μηχανικής εκμάθησης που λειτουργούν σε μικροελεγκτές υπό συνθήκες χαμηλής ισχύος. Σχεδιάσαμε και εκπαιδεύσαμε αρκετά μοντέλα ανίχνευσης λέξεων-κλειδιών, βελτιστοποιημένα τους για καλύτερη επίδοση πάνω στη συσκευή και τα συγκρίναμε με μοντέλα που έχουν εκπαιδευτεί με εμπορικό λογισμικό από άποψη ακρίβειας, επεργαστικού χρόνου και μεγέθους .Επίσης ενσωματώθηκε επικοινωνία BLE έτσι ώστε αν βρεθούν οι λέξεις τότε οι τιμές των αισθητήρων να αποστέλλονται στην εφαρμογή ενός smartphone. Τέλος, αξιολογήθηκε η απόδοση της εφαρμογής εκτελώντας την σε διαφορετικές αρχιτεκτονικές και μικροεπεξεργαστές και υπολογίστηκε η κατανάλωση ενέργειας για τον μικροελεγκτή nrf52840.

Λέξεις κλειδιά

Βαθιά Μηχανική Μάθηση,Ενσωματωμένες συσκευές ,Αναγνώριση Λέξεων,Λειτουργικό Πραγματικού Χρόνου,Μηχανική Μάθηση σε Μικροελεγκτές

Abstract

A keyword spotting system implemented on an embedded device using convolutional neural networks .The main focus of this thesis was the design,implementation and evaluation of Machine Learning models running on microcontrollers under low-power conditions .We designed and trained several keyword detection models ,optimized them for on-device deployment and compared them against models trained with commercial software in terms of accuracy,inference speed and size.We also implemented BLE communication so that if the words are found sensor values are being sent to a smartphone . Finally we compared the performance of the application by running it on different architectures and microprocessors and evaluated the power consumption for our nrf52840 microcontroller .

Keywords

Keyword spotting, Speech recognition, Embedded Machine Learning, Deep learning, Convolutional neural networks, Quantization,TinyML,RTOS,

Ευχαριστίες

Η εκπόνηση της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε στα πλαίσια των προπτυχιακών σπουδών μου στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Πριν προχωρήσω στην περιγραφή της εργασίας και των αποτελεσμάτων που προέκυψαν, θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους που συνέβαλαν στην ολοκλήρωση της συγκεκριμένης εργασίας.

Αρχικά θα ήθελα να απευθύνω τις ευχαριστίες μου στον καθηγητή του Ε.Μ.Π. κύριο Παναγιώτη Τσανάκα, για την δυνατότητα που μου προσέφερε να εκπονήσω τη διπλωματική μου εργασία πάνω σε ένα ιδιαίτερα ενδιαφέρον αντικείμενο και να διευρύνω με αυτό το τρόπο τις επιστημονικές και τεχνολογικές μου γνώσεις. Επιπλέον θα ήθελα να ευχαριστήσω τον Θοδωρή ,τον Παύλο και τον Θανάση για την βοήθειά τους στην υλοποίηση της εργασίας μου .

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου, η οποία στάθηκε δίπλα μου και με στήριξε όλα αυτά τα χρόνια, καθώς και τους φίλους και συμφοιτητές μου, οι οποίοι μέσω της συνεργασίας μας συνέβαλαν στην ολοκλήρωση της ακαδημαϊκής μου σταδιοδρομίας.

Αθήνα, Οκτώβριος 2021

Δημήτριος Κόσυβας

Πίνακας Περιεχομένων

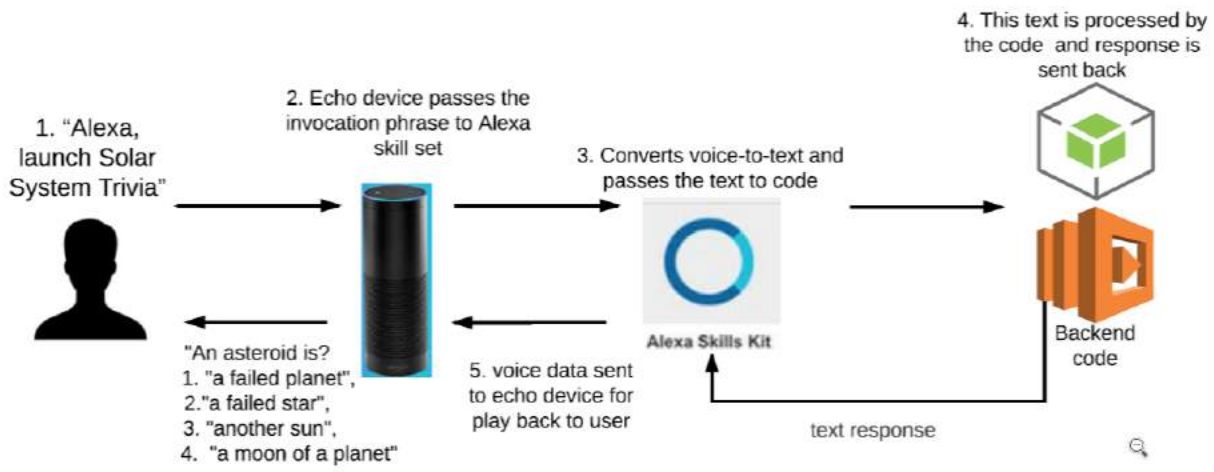
1.Εισαγωγή

1.1 Κίνητρο

Την τελευταία δεκαετία οι αλγόριθμοι βαθιάς μηχανικής μάθησης βελτιώνουν συνεχώς την επίδοση ενός μεγάλου εύρους εφαρμογών μεταξύ των οποίων και της αυτόματης αναγνώρισης φωνής[1].Έτσι παρατηρείται αύξηση στην παραγωγή συσκευών αναγνώρισης φωνής σαν και αυτές που φαίνονται στο παρακάτω σχήμα .

Εικόνα 1.1*:Voice-controlled devices*

Ωστόσο η αναγνώριση ανθρώπινης ομιλίας σε αυτές τις ηλεκτρονικές συσκευές στηρίζεται σε natural language processing (NLP) ,το οποίο συνήθως απαιτεί πρόσβαση σε cloud για τις βαριές υπολογιστικά διεργασίες .Όμως λύσεις εξαρτώμενες από το cloud δεν είναι πρακτικές για πολλές συσκευές καθώς χρειάζονται πρόσβαση στο internet για να λειτουργήσουν και αυτή η συνεχόμενη μετάδοση ήχου κρύβει προβλήματα ασφάλειας για τους χρήστες . Το pipeline για το παράδειγμα της Alexa στην εικόνα 1.2 δείχνει τα στάδια μετάβασης της πληροφορίας και πλέον είναι φανερό ότι η καθυστέρηση του συστήματος εκτός από την ταχύτητα της σύνδεσης εξαρτάται και από τον αριθμό των blocks επεξεργασίας [2].



Εικόνα 1.2*:Information flow in Alexa*

Στα προβλήματα αυτά δίνει λύση η αναγνώριση λέξεων σε μικροελεγκτές , όπου όλη επεξεργασία του ήχου γίνεται στη συσκευή και ανάλογα με την εύρεση της λέξης εκτελούνται αντίστοιχες ενέργειες χρησιμοποιώντας τα περιφερειακά .

1.2 Προσέγγιση Προβλήματος

Τα νευρωνικά δίκτυα επιλέχθηκαν για την επίλυση του στόχου δηλαδή την δημιουργία μοντέλου αναγνώρισης λέξεων σε ενσωματωμένη συσκευή .Για αυτό το λόγο αρχικά έγινε έρευνα των διαφορετικών Machine Learning Frameworks όπως scikit-learn, Keras, Caffe and TensorFlow το οποία δίνουν την δυνατότητα της μετατροπής των εκπαιδευμένων μοντέλων σε μορφή συμβατή για μικροελεγκτές ,οι οποίοι συνήθως προγραμματίζονται με C/C++ .Αυτά τα frameworks αναλαμβάνουν το κομμάτι του νευρωνικού και επιτρέπουν στους χρήστες να ασχοληθούν αποκλειστικά με το κώδικα της εφαρμογής. Τα τελευταία χρόνια έχουν εμφανιστεί open-source και εταιρικά frameworks ειδικά για την εκτέλεση προεκπαιδευμένων μοντέλων σε μικροελεγκτές όπως το Edge Impulse,Tensorflow Lite ,X-CUBE-AI .Το X-CUBE-AI [3] είναι ένα εργαλείο της STMicroelectronics που μετατρέπει το προεκπαιδευμένο μοντέλο από συγκεκριμένα Deep Learning frameworks σε μια βελτιστοποιημένη βιβλιοθήκη. Ωστόσο το X-CUBE-AI μπορεί να χρησιμοποιηθεί μόνο σε STM32 μικροελεγκτές και δεν είναι open-source .
Σε αυτή την εργασία για την εφαρμογή αναγνώρισης έγινε χρήση του Tensorflow Lite και του Edge Impulse .

1.3 Στόχος

Ο σκοπός της διπλωματικής είναι η ανάπτυξη συστήματος αναγνώρισης λέξεων σε μικροελεγκτή ,το οποία με την εύρεση της αντίστοιχης λέξης θα επιστρέφει τιμές αισθητήρων μέσω BLE σε smartphone .
Για αυτό το λόγο έγιναν τα παρακάτω :

- Εκπαίδευση νευρωνικού δικτύου για την αναγνώριση συγκεκριμένων λέξεων
- Υλοποίηση on device inference για τον nrf52840 χρησιμοποιώντας Tensorflow Lite μέσω Zephyr RTOS.
- Σύγκριση της επίδοσης του Tensorflow Lite μοντέλου με την υλοποίηση μέσω από τον Edge Impulse.
- Δημιουργία συστήματος για την δειγματοληψία του αισθητήρα και την αποστολή των τιμών τους μέσω Bluetooth Low Energy.
- Εκτέλεση της εφαρμογής σε άλλες ενσωματωμένες συσκευές για την σύγκριση των χρόνων επεξεργασίας και την κατανάλωση σε μνήμη.
- Ανάλυση της κατανάλωσης ισχύος της εφαρμογής στον μικροελεγκτή nrf52840 .

2.Θεωρητικό υπόβαθρο

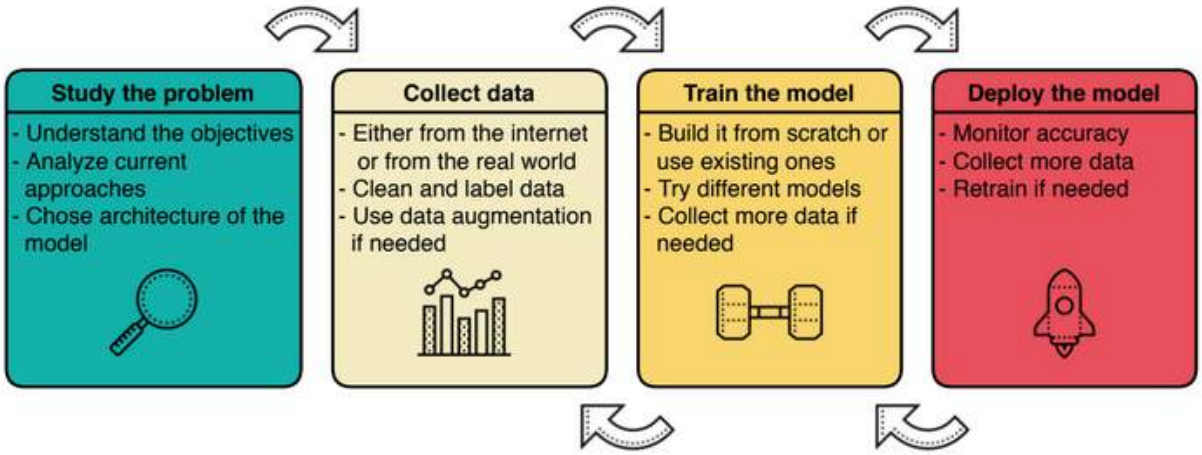
2.1 Μηχανική Μάθηση

Σύμφωνα με τον Arthur Samuel [4],η μηχανική μάθηση είναι η επιστήμη η οποία δίνει την δυνατότητα στους υπολογιστές να μαθαίνουν μόνοι τους και να οδηγούνται στη λύση του προβλήματος .Συνεπώς οι αλγόριθμοι μηχανικής μάθησης εκπαιδεύονται πάνω σε μια σειρά παραδειγμάτων που περιγράφουν ένα φαινόμενο [5].Αυτές οι συλλογές παραδειγμάτων ονομάζονται σύνολα δεδομένων (datasets) και μπορεί να παραχθούν είτε φυσικά είτε τεχνητά.

Για να κατανοήσουμε πως μπορούν να λυθούν προβλήματα καλύτερα χρησιμοποιώντας μηχανική μάθηση θα εξετάσουμε μία απλή εφαρμογή . Έστω ότι χρειάζεται να δημιουργηθεί ένα σύστημα που προβλέπει την κίνηση των ανθρώπων με βάση τα δεδομένα ενός επιταχυνσιομέτρου.Το μοντέλο θα εκπαιδευτεί σε ένα dataset με μετρήσεις επιταχυνσιομέτρου σε διαφορετικές κατηγορίες κινήσεων όπως περπάτημα,τρέξιμο,χορός,ακινησία. Η είσοδος στο σύστημα μπορεί να είναι είτε οι καθαρές μετρήσεις και των τριών αξόνων είτε στοιχεία που προέκυψαν από την επεξεργασία των μετρήσεων όπως RMS,φασματική ισχύς ,peak frequency , peak amplitude .Αυτά είναι γνωστά ως χαρακτηριστικά ,τα οποία περιγράφουν το φαινόμενο που παρατηρείται . Η έξοδος του συστήματος θα είναι κάποια πρόβλεψη κίνησης ,η οποία παράγεται από τον εκπαιδευμένο μοντέλο .Για την αξιολόγηση του μοντέλου ελέγχουμε την ακρίβεια, δηλαδή το ποσοστό ορθών προβλέψεων για τα δεδομένα εισόδου .Υπάρχει μεγάλη ποικιλία αλγορίθμων μάθησης ,οι οποίοι διαχωρίζονται με διάφορους τρόπους και ένας από αυτούς εξαρτάται από την επίβλεψής τους (supervision) κατά την εκπαίδευση .Οι αλγόριθμοι K-nearest-neighbours,γραμμική και λογιστική παλινδρόμηση ,οι Μηχανές Διανυσμάτων Υποστήριξης (SVM) βρίσκονται στη κατηγορία της επιβλεπόμενης μάθησης .Έτσι τροφοδοτούνται με δεδομένα που περιέχουν λύσεις (labels) όπως το παραπάνω παράδειγμα .Ενώ αλγόριθμοι όπως οι k-Means,expectation-maximization εντάσσονται στη κατηγορία της μη επιβλεπόμενης μάθησης ,όπου δέχονται δεδομένα εκπαίδευσης χωρίς ετικέτες (labels) και προσπαθούν να βρουν ομοιότητες μεταξύ τους .Επίσης υπάρχουν και άλλες κατηγορίες μάθησης όπως ημί-επιβλεπόμενη και ενισχυτική μάθηση. Τα νευρωνικά δίκτυα εμπνεύστηκαν από τους νευρώνες του ανθρώπινου εγκεφάλου και μπορούν να καταταχθούν σε οποιαδήποτε από τις παραπάνω κατηγορίες .Παρατηρούμε την εφαρμογή τους σε σύνθετα προβλήματα αναγνώρισης εικόνων , ήχων και στην αυτόματη οδήγηση ,αλλά απαιτούν μεγάλο όγκο δεδομένων και ισχυρή υπολογιστική δύναμη για την εκπαίδευσή τους .Η διαδικασία όπου το προ-εκπαιδευμένο μοντέλο δέχεται κάποια είσοδο και επιστρέφει προβλεψη λέγεται inference .Επειδή η εκτέλεση της απαιτεί λιγότερους υπολογιστικούς πόρους από την εκπαίδευση μπορεί να τρέχει σε smartphones,tablets internet browsers ,μικροϋπολογιστές αλλά και μικροελεγκτές όπως θα αναλύσουμε στα επόμενα κεφάλαια .

2.1.1 Γενική Ροή ML

Υπάρχουν συγκεκριμένα βήματα στην ροή της ΜΜ ,ώστε από μία ιδέα να προκύψει ένα λειτουργικό μοντέλο τα οποία φαίνονται παρακάτω .



***Εικόνα 2.1**: Workflow diagram of solving a generic Machine Learning problem*

Αρχικά μελετάται το πρόβλημα ,για να κατανοηθούν οι στόχοι ,να βρεθούν οι ήδη υπάρχων λύσεις και να αποφασιστεί η προσέγγιση που θα χρησιμοποιηθεί.Στη συνέχεια επιλέγουμε μία πρωτογενή αρχιτεκτονική του μοντέλου ανάλογα το είδος του προβλήματος .Το επόμενο βήμα είναι η συλλογή δεδομένων είτε από πηγές στο internet είτε δημιουργώντας τα δικά μας δεδομένα δειγματοληπτώντας φαινόμενα από το πραγματικό κόσμο .Καθώς η διαδικασία της συλλογής είναι συνήθως δύσκολη και χρονοβόρα μπορούμε να χρησιμοποιήσουμε μεθόδους μεγιστοποίησης του όγκου και της ποικιλίας των δεδομένων (data augmentation techniques) .Στο τρίτο στάδιο εκπαιδεύουμε το μοντέλο σε διαφορετικές αρχιτεκτονικές και επιλέγουμε αυτή που συμπεριφέρεται καλύτερα .Για να βελτιώσουμε την ακρίβεια του μοντέλου επαναλαμβάνεται αρκετές φορές το τρίτο βήμα .Τέλος ανεβάζουμε το μοντέλο στη συσκευή που επιθυμούμε και ελέγχουμε εκεί την πραγματική ακρίβεια του μοντέλου.Εάν η επίδοση δεν μας ικανοποιεί μπορούμε πάντα να συλλέξουμε νέα δεδομένα και να επανεκπαιδεύσουμε το μοντέλο.

2.1.2 TinyML

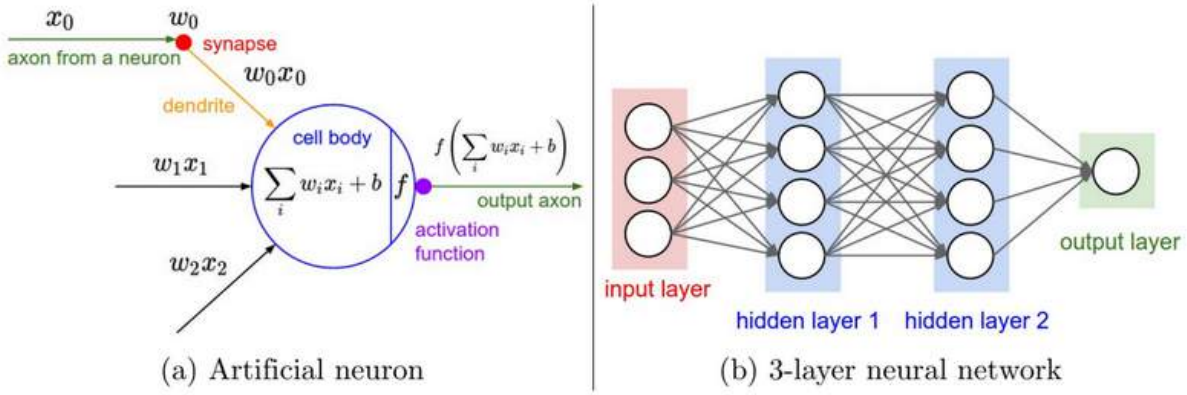
Η χρήση μηχανική μάθησης σε ενσωματωμένες συσκευές ή αλλιώς TinyML είναι ένας ανερχόμενος τομέας ,που συνδέεται άμεσα με το Internet of Things.Ωστόσο παρά την άνοδό του παρατηρείται ότι υπάρχουν λίγοι πόροι σε σχέση με τα αμέτρητα εργαλεία που υποστηρίζουν τη Μηχανική Μάθηση σε υπολογιστές ή servers .Έτσι οι πιο πολλές πληροφορίες αντλήθηκαν από επιστημονικά άρθρα ,δημοσιεύσεις σε blog και machine learning framework documentation [6] [7] [8] [9] [10]. Ένα από τα σημαντικότερα πλεονεκτήματα εκτέλεσης αλγορίθμων ML σε μικροελεγκτές είναι η ελάχιστη κατανάλωση ισχύος ,καθώς στις περισσότερες IoT εφαρμογές υπάρχουν συσκευές που στέλνουν καθαρές τιμές αισθητήρων μέσω ενός ασύρματου δικτύου σε ένα server.Συνεπώς η επεξεργασία τους γίνεται εξ' ολοκλήρου στο server και η χρήσιμη πληροφορία στέλνεται πίσω στη συσκευή. Όμως η ασύρματη επικοινωνία για τις ενσωματωμένες συσκευές έχει μεγαλύτερο ενεργειακό κόστος σε σχέση με τον απλούς υπολογισμούς on device .Για παράδειγμα η επικοινωνία μέσω Bluetooth μπορεί να καταναλώσει μέχρι 100 miliwatts ενώ to MobileNetV2 δίκτυο για αναγνώριση εικόνων με 1 inference ανά sec καταναλώνει μέχρι 110 microwatts .Καθώς οι συσκευές αυτές συνήθως τροφοδοτούνται από μπαταρία είναι καθοριστικό να ελαχιστοποιούνται τα δεδομένα που στέλνονται ασύρματα. Επομένως αντί να στέλνουν ότι καταγράφουν είναι πολύ πιο αποδοτικό να γίνεται η επεξεργασία στη συσκευή και να αποστέλλονται μόνο τα αποτελέσματα .

Η μειωμένη καθυστέρηση αποτελεί έναν επιπλέον λόγο να χρησιμοποιήσουμε ML σε low-power συσκευές .Αφού αν καταφέρουμε να εξάγουμε όλη τη χρήσιμη πληροφορίες από τα αρχικά δεδομένα πάνω στη συσκευή ,ο τελικός χρόνος αποστολής των αποτελεσμάτων αλλάζει από seconds σε milliseconds

Ωστόσο υπάρχουν και μειονεκτήματα στην εκτέλεση νευρωνικών δικτύων σε ενσωματωμένες συσκευές καθώς εξαιτίας των περιορισμένων πόρων η εκπαίδευση των μοντέλων πάνω σε αυτές είναι αδύνατη .Επίσης το μέγεθος του μοντέλου πρέπει να είναι μικρό ώστε να χωράει στη μνήμη του μικροελεγκτή ,η οποία στους περισσότερους ξεκινάει από μερικές εκατοντάδες kilobyte έως 2 megabyte .

2.2 Νευρωνικά Δίκτυα

Τα πρώτα μοντέλα νευρωνικών δικτύων εμφανίστηκαν στο 1943 [4](McCulloch και Pitts) και λειτούργησαν ως τους πρωτεργάτες για την τεχνητή νοημοσύνη .Ωστόσο χρειάστηκε να περάσουν κάποιες δεκαετίες έρευνας και προόδου ώστε να μπορέσουν να εφαρμοστούν σε πρακτικά καθημερινά προβλήματα .Η δημιουργία τους εμπνεύστηκε από το τρόπο λειτουργίας των βιολογικών νευρωνικών συστημάτων .Οι McColloc και Pitts κατάφεραν να αποδείξουν ότι ένα απλό μοντέλο τεχνητού νευρωνικού δικτύου με μία η περισσότερες δυαδικές εισόδους και μία δυαδική έξοδο μπορεί να υπολογίσει μια λογική πρόταση ,λειτουργώντας ως μέλος ενός μεγαλύτερου δικτύου.

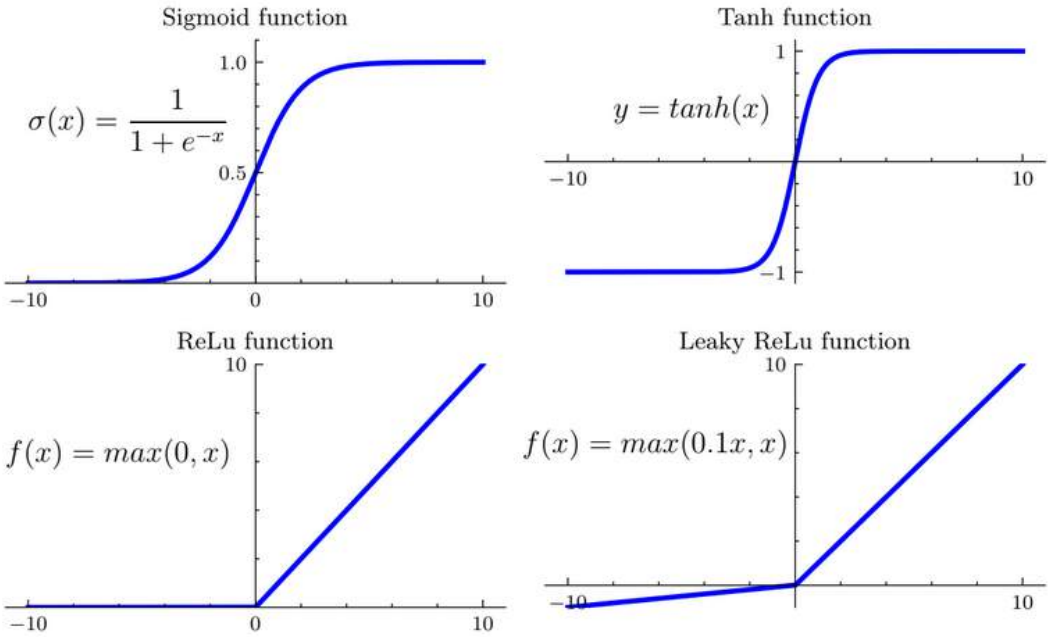


Εικόνα 2.2 *:a)Mathematical model of artificial neuron ,b) fully-connected 3-layer network*

Όπως φαίνεται στο παραπάνω σχήμα 2.2 a) ένας νευρώνας μπορεί να πάρει διάφορες εισόδους ,πολλαπλασιάζει κάθε είσοδο με το βάρος της και τα αθροίζει . Στη συνέχεια προσθέτουμε μία σταθερά στο άθροισμα και οδηγείται στην συνάρτηση ενεργοποίησης. Τα νευρωνικά δίκτυα αποτελούνται από πολλούς νευρώνες ,οι οποίοι οργανώνονται σε στρώματα (layers) και διασυνδέονται μόνο με άλλους νευρώνες γειτονικών στρωμάτων Το πρώτο στρώμα ορίζεται ως είσοδος , όλα τα ενδιάμεσα ως κρυφά (hidden) και το τελευταίο ως έξοδος . Αν όλες οι είσοδοι των νευρώνων σε ένα στρώμα συνδέονται με όλες τις εξόδους του προηγούμενου τότε το στρώμα λέγεται πλήρες συνδεδεμένο (fully connected ή dense) ,όπως φαίνεται στο σχήμα 2.2 b) .Γενικότερα τα νευρωνικά δίκτυα με πολλά κρυφά στρώματα ανήκουν στην κατηγορία DNN (Deep Neural Networks) .

2.2.1 Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης (activation functions) εισάγουν τη μη γραμμικότητα στο σύστημα ,που επιτρέπει στο νευρωνικό δίκτυο να προσεγγίζει τις συνεχείς συναρτήσεις.Υπάρχουν πολλά διαφορετικά είδη συναρτήσεων όπως η σιγμοειδής και η rectified linear activation function (ReLu) .Στο παρελθόν έγινε χρήση της σιγμοειδής για την προσομοίωση της λειτουργίας του βιολογικού νευρώνα ,όπου δέχεται έναν πραγματικό αριθμό και τον συμπιέζει σε έυρος 0 και 1 .Ωστόσο στη συνέχεια αποδείχθηκε ότι η εκπαίδευση νευρωνικών δικτύων με τη σιγμοειδή δυσχαιρένεται καθώς οι κορεσμένες έξοδοι αφαιρούν κομμάτια του δικτύου [11].Έτσι ο αλγόριθμος μάθησης δεν φτάνει σε όλους τους νευρώνες με αποτέλεσμα να προκύπτει σφάλμα στα βάρη .Για το λόγο αυτό πλέον έχει αντικατασταθεί από την ReLu].



***Εικόνα 2.3**:Different Activation functions and their equations*

Η softmax είναι μία άλλη συνάρτηση ενεργοποίησης ,όπου όπως φαίνεται στη 2.1 δέχεται σαν είσοδο ένα διάνυσμα ,υπολογίζει το εκθετικό του κάθε στοιχείου και διαιρείται με το άθροισμα όλων των εκθετικών από τα στοιχεία του διανύσματος .Το διάνυσμα τιμών εισόδου γίνεται διάνυσμα πιθανοτήτων και χρησιμοποιείται συνήθως ως το τελευταίο στρώμα σε ένα νευρωνικό .

$$\sigma(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}} \quad \text{for } i = 1, \cdots, K \text{ and } y = (y_1, \cdots, y_k) \in \mathbb{R}^K \quad (2.1)$$

Where:

y - Input vector

K - Number of elements in the input vector

$\sigma(y_i)$ - Computed probability of the i-th element in the input vector

2.2.2 Backpropagation

Ο αλγόριθμος που χρησιμοποιείται για την εκπαίδευση των νευρωνικών λέγεται **backprogation** .Όπως αναφέρθηκε χρειαζόμαστε έναν μεγάλο αριθμό δεδομένων ταξινομημένα σε κλάσεις για να εκπαιδεύσουμε το μοντέλο.Στην αρχή της εκπαίδευσης όλα τα βάρη και οι σταθερές ορίζονται σε τυχαίες τιμές .Κατά τη διάρκεια κάθε βήματος εκπαίδευσης παρέχεται στο μοντέλό ένα μικρό κομμάτι δεδομένων και υπολογίζονται οι αντίστοιχες καρτέλες εξόδου .Η διαδικασία αυτή λέγεται **forward pass** και αποθηκεύει τα ενδιάμεσα αποτελέσματα από όλων των νευρώνων κάθε στρώματος .Στη συνέχεια κάθε έξοδος συγκρίνεται με την αναμενόμενη τιμή χρησιμοποιώντας μια συνάρτηση κόστους, η οποία μας ενημερώνει για την επίδοση του μοντέλου .Όσο μεγαλύτερη τιμή επιστρέφει τόσο χειρότερα συμπεριφέρεται το νευρωνικό,οπότε ο στόχος είναι η ελαχιστοποίηση της συνάρτησης κόστους για να αυξήσουμε την ακρίβεια του μοντέλου.Αυτό σημαίνει ότι πρέπει να υπολογιστεί η μεταβολή για τα βάρη και τις σταθερές ,ωστέ να μειωθεί η έξοδος της συνάρτησης.

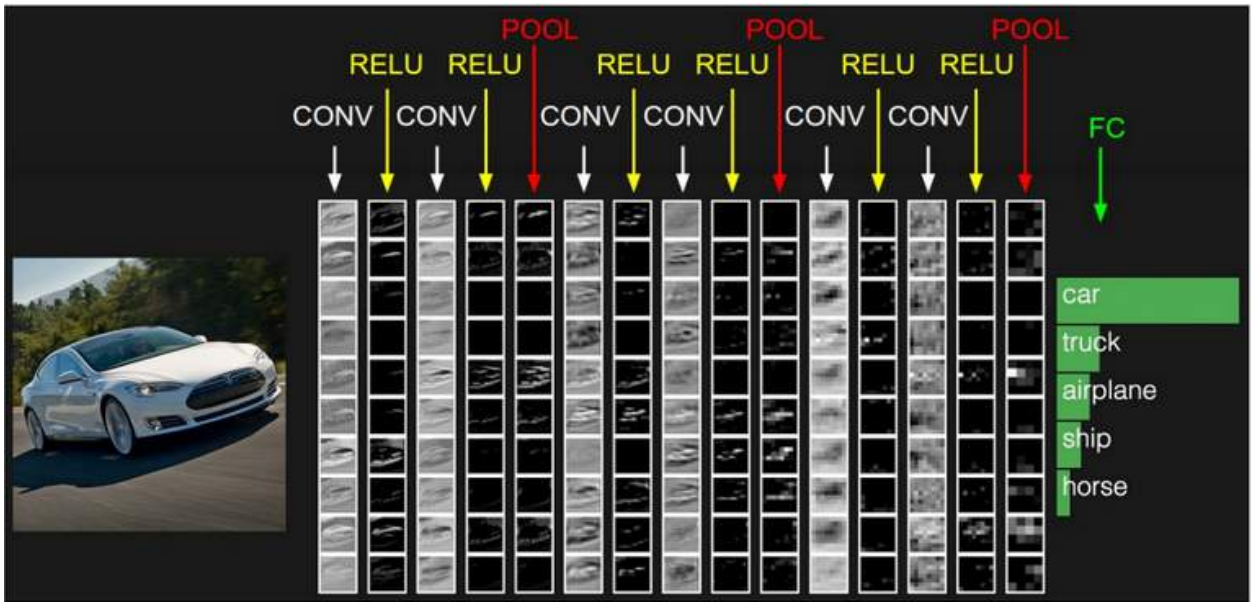
Η εύρεσης των παραμέτρων χρησιμοποιώντας τον αλγόριθμο backpropagation γίνεται σε βήματα .Αφού υπολογιστεί η συνάρτηση κόστους ,ο αλγόριθμος βρίσκει αναλυτικά πόσο συνείσφερε κάθε έξοδος σε αυτή ,με την βοήθεια των ενδιάμεσων αποθηκευμένων τιμών .Αυτό το βήμα επαναλαμβάνεται αναδρομικά για κάθε στρώμα μέχρι να φτάσει στην είσοδο του νευρωνικού ,όπου πλέον ο αλγόριθμος γνωρίζει σε ποια κατεύθυνση πρέπει να μεταβληθούν οι παράμετροι για να μειωθεί το κόστος . Η διαδικασία ονομάζεται **Gradient Descent** ,όπου όλες οι τοπικές κλίσεις πολλαπλασιάζονται με το **learning rate** και αφαιρούνται από τα βάρη και τις σταθερές .

Επιπλέον δεν χρειάζεται να εκτελεστεί backpropagation για κάθε στιγμιότυπο εκπαίδευσης ,αλλά μπορούμε να κάνουμε προβλέψεις για ένα μικρότερο dataset ,να υπολογίσουμε τη μέση συνάρτηση κόστους και μετά να εφαρμόσουμε τον αλγόριθμο .

2.2.3 Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (CNN) είναι ένα είδος νευρωνικών ,χρησιμοποιούνται ιδιαίτερα για κατηγοριοποίηση εικόνων .Η έμπνευση για την δημιουργία τους προήλθε από τη συμπεριφορά του οπτικού φλοιού του εγκεφάλου σε διαφορετικά ερεθίσματα .

Στο σχήμα 2.4 βλέπουμε ένα παράδειγμα CNN ,το οποίο δέχεται την εικόνα ενός αυτοκινήτου ως είσοδο και επιστρέφει τις πιθανότητες κατηγοριοποίησης για τις πέντε διαφορετικές κλάσεις .



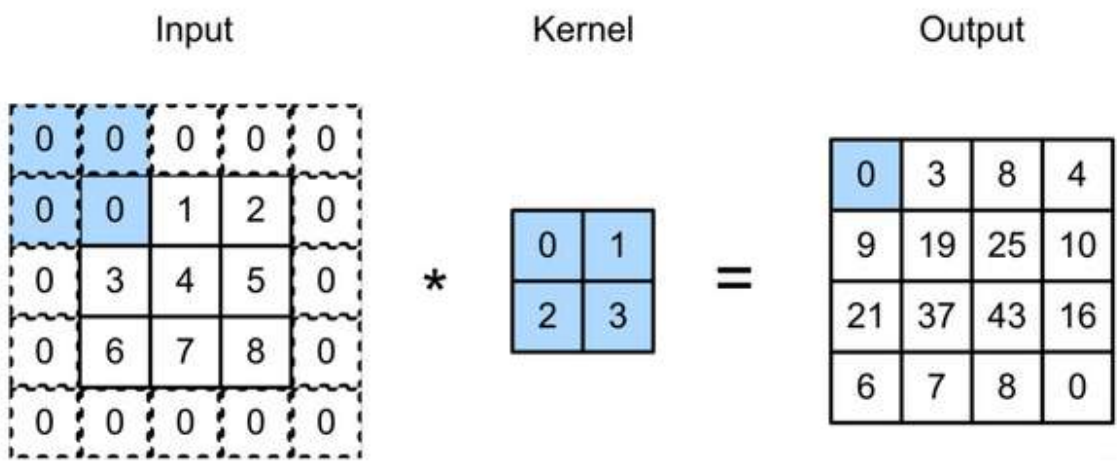
***Εικόνα 2.4**:Structure of a CNN [11]*

Τα CNN διαθέτουν δύο διαφορετικά είδη στρωμάτων , **convolutional** και **pooling** .Κάθε convolutional στρώμα ανιχνεύει κάποιο είδος σχημάτων : τα πρώτα βρίσκουν διαφορετικά είδη γωνιών ,ενώ τα τελευταία στο νευρωνικό διακρίνουν πιο σύνθετα αντικείμενα και σχήματα όπως ρόδες ,πόδια ,μάτια ,αυτιά .Ενώ τα pooling στρώματα υποδειγματοληπτούν τα δεδομένα σε χωρική διάσταση ,με αποτέλεσμα την μείωση των παραμέτρων και της πολυπλοκότητας των πράξεων στο νευρωνικό. Μετά από ορισμένα εναλλασσόμενα ζεύγη από convolutional και polling στρωμάτων ,παράγεται η έξοδος από το τελευταίο pooling στρώμα και μετατρέπεται σε ένα μονοδιάστατο διάνυσμα το οποίο δίνεται ως είσοδος στο πλήρες συνδεδεμένο νευρωνικό δίκτυο.Τελικά από αυτό προκύπτουν οι πιθανότητες για όλες τις κλάσεις των δεδομένων όπως φαίνεται στην εικόνα 2.4.

2.2.3.1 Convolutions Layers

Τα CNN λειτουργούν με τρισδιάστατα δεδομένα (tensors) ,όπου το πλάτος και το ύψος περιγράφουν την ανάλυση της εικόνας και το βάθος αντιστοιχεί στον αριθμό των καναλιών 3 για RGP (red,green,blue) και 1 για το ασπρόμαυρο.

Αυτά τα συνελικτικά στρώματα υπολογίζουν το εσωτερικό γινόμενο μεταξύ διανυσμάτων εισόδου και φίλτρων και παράγουν μία έξοδο . Οι παράμετροι των φίλτρων ορίζονται στο στάδιο της εκπαίδευσης .Στο παρακάτω σχήμα 2.5 φαίνεται ένα 2d φίλτρο 2x2 ,το οποίο πολλαπλασιάζεται με το διάνυσμα εισόδου και προκύπτει η έξοδος .



***Εικόνα 2.5**:Dot product operation between filter and zero-padded input matrix. [12]*

Το μέγεθος των διαστάσεων (ύψος & πλάτος) της εξόδου εξαρτάται από διάφορες παραμέτρους όπως φαίνεται στη 2.2 .

$$V_o = (V_i - F + 2P)/S + 1 \tag{2.2}$$

Where:

V_i - Input volume size, only width or height

V_o - Output volume size, only width or height

F - Filter or receptive field size

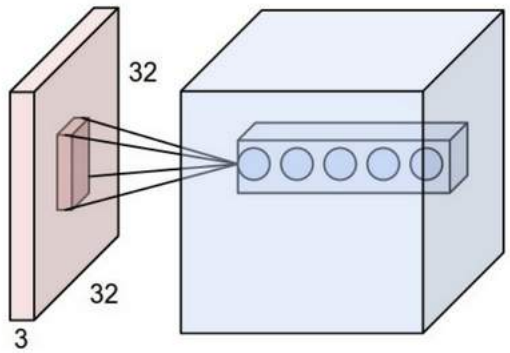
P - Amount of zero padding used on the border

S - Stride length

Αν εξετάσουμε το παράδειγμα στο σχήμα 2.5 βλέπουμε ότι η είσοδος είναι ένας πίνακας 3x3 με stride 1 ,padding 1 και εφαρμόζεται φίλτρο 2x2 για να προκύψει μία έξοδος μεγέθους 4x4 .

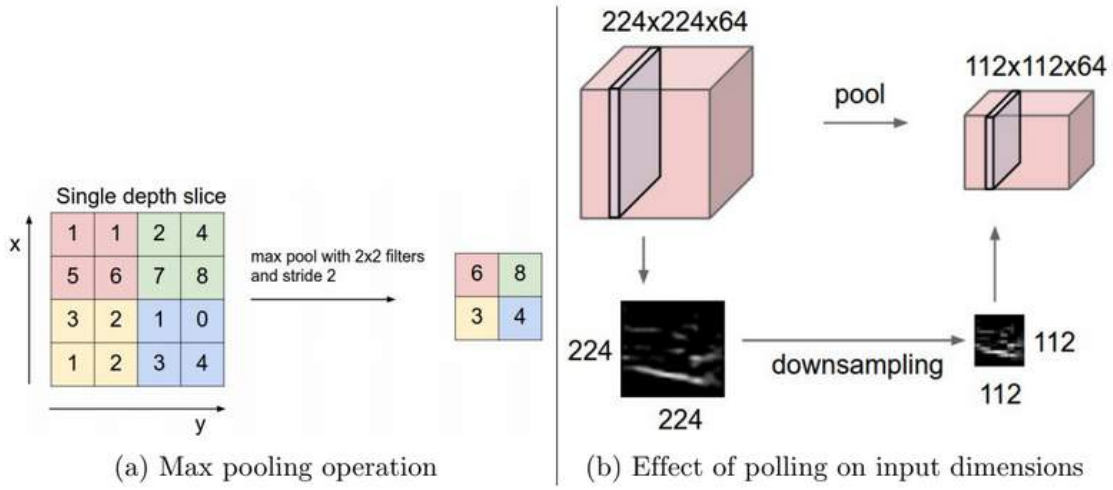
Το βάθος της εξόδου ισούται με τον αριθμό των φίλτρων που χρησιμοποιήθηκαν και στο τέλος του convolutional στρώματος η έξοδος διοχετεύεται σε άλλους νευρώνες . Ολα τα στοιχεία που βρίσκονται στο ίδιο βάθος επηρεάζονται από τον ίδιο σταθερό παράγοντα και τροφοδοτούνται στη συνάρτηση ενεργοποίησης .

2.2.3.2 Pooling Στρώματα



***Εικόνα 2.6**:Convolutional layer with five different filters [12]*

Τα pooling στρώματα εκτελούν υποδειγματοληψία (downsampling) του ύψους και πλάτους της εισόδου .Αυτό πραγματοποιείται ολισθαίνοντας ένα φίλτρο σταθερού μεγέθους πάνω στην είσοδο και εφαρμόζοντας τη MAX διαδικασία σε όλα τα στοιχεία που καλύπτει το φίλτρο ,ώστε μόνο τα μεγαλύτερα στοιχεία να αντιγράφονται στην έξοδο .Το pooling γίνεται για κάθε βάθος τεμαχίου ξεχωριστά από τα άλλα ,οπότε το βάθος διατηρείται σταθερό .



Εικόνα 2.7*:Pooling layer*

Όπως φαίνεται στη εικόνα 2.7.a η επιλογή των διαστάσεων 2x2 για pooling έχει ως αποτέλεσμα την μεταβολή ύψους και πλάτους της εισόδους στο μισό ,αγνοώντας το 75% των ενεργοποιήσεων. Επομένως τα pooling στρώματα χρησιμοποιούνται γιατί μειώνουν τις παραμέτρους του νευρωνικού και άρα την απαιτούμενη μνήμη και το σύνολο των υπολογισμών .

2.3 Tensorflow

Το Tensorflow είναι ένα δωρεάν open-source framework για αριθμητικούς υπολογισμούς και ιδανικό για εφαρμογές μηχανικής μάθησης .Αρχικά ξεκίνησε ως εταιρικό project από την ομάδα Google Brain το 2011 και αργότερα το 2015 έγινε open-source .Η Google το χρησιμοποιεί σε αρκετά προϊόντα της όπως το Gmail,Google Cloud Speech και Google Search.Παρέχει στους προγραμματιστές τα εργαλεία για την δημιουργία και εκπαίδευση μοντέλων ML ,χωρίς να χρειάζεται βαθιά γνώση της θεωρίας των νευρωνικών δικτύων . Συνεπώς οι προγραμματιστές γράφουν κώδικα στο Python API,το οποίο καλεί βελτιστοποιημένο C++ κώδικα . Όταν χρησιμοποιούμε Tensorflow το δυσκολότερο κομμάτι είναι συνήθως η προετοιμασία των δεδομένων.Μόλις γίνει αυτό η δημιουργία ,εκπαίδευση και αξιολογηση του μοντέλου πραγματοποιείται με λίγες γραμμές κώδικα σε Python . Ακόμα υποστηρίζει το Keras API για την κατασκευή ML μοντέλων ,το οποίο είναι μια python βιβλιοθήκη που λειτουργεί ως “wrapper” για το tensorflow .Έτσι οι χρήστες δεν χρειάζεται να ασχολούνται με τις διασυνδέσεις μεταξύ των στρωμάτων ,αλλά πρέπει μόνο να διαλέγουν τον κατάλληλο τύπο στρώματος (convolutional,max pool,fully connected) και το μέγεθός του.

Ένα σημαντικό πλεονέκτημα χρήσης του tensorflow είναι ότι η μορφή του εξαγόμενου εκπαιδευμένου μοντέλου είναι συμβατή για διαφορετικά περιβάλλοντα .Έτσι το μοντέλο μπορεί να εκπαιδευτεί σε python σε ένα μηχάνημα Linux και να εκτελεστεί με Java για μία android συσκευή .Αυτή η ιδιότητα είναι πολύ σημαντική για την εφαρμογή ML μοντέλων σε μικροελεγκτές.

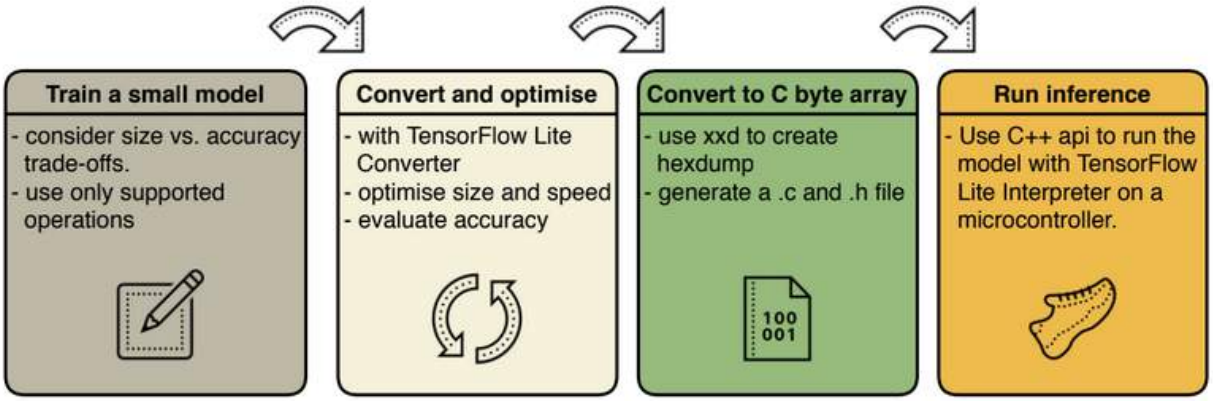
2.3.1 Tensorflow Lite για Μικροελεγκτές

Το tensorflow-lite (TFLite) είναι μία σειρά εργαλείων και βιβλιοθηκών ,που επιτρέπουν την εκτέλεση ML inference σε low power συσκευές .Υπάρχει υποστήριξη και documentation για Android ,iOS συσκευές και embedded Linux .Το πακέτο tensorflow-lite για μικροελεγκτές (TFLite Micro) προστέθηκε στο TFLite το 2019 και χρησιμοποιεί το C++ API ,το οποίο αποτελεί μεγάλο μέρος του πυρήνα της tensorflow .

Η TFLite Micro βιβλιοθήκη δεν έχει εξαρτήσεις από συγκεκριμένα περιφερειακά ,οπότε ο ίδιος C++ κώδικας μπορεί να μεταγλωτιστεί για να τρέχει σε μικροελεγκτή ή σε υπολογιστή με λίγες αλλαγές . Ωστόσο οι χρήστες πρέπει να υλοποιήσουν την δικιά τους έκδοση της printf() συνάρτηση ,ανάλογα με το μικροελεγκτή που χρησιμοποιούν .

Η βιβλιοθήκη είναι διαθέσιμη στο Github ως μέρος του tensorflow project και αν κάποιος θέλει να τη χρησιμοποιήσει σε ενσωματωμένες συσκευές θα πρέπει να κάνει clone ολόκληρο το tensorflow [13] .Η ομάδα του tensorflow παρέχει κάποια παραδείγματα σε διαφορετικές πλατφόρμες όπως Mbed OS,Arduino,OpenMV και ESP32.

Τα γενικά βήματα για την δημιουργία ML μοντέλου που αναλύσαμε προηγουμένως και φαίνονται στην εικόνα 2.1 συνεχίζουν να ισχύουν.Ωστόσο υπάρχουν κάποια επιπλέον βήματα για την εκτέλεση μοντέλων μηχανική μάθησης σε low power συσκευές ,όπως φαίνονται στη 2.8 .



***Εικόνα 2.8**:Workflow of preparing a ML model for an inference on a microcontroller.*

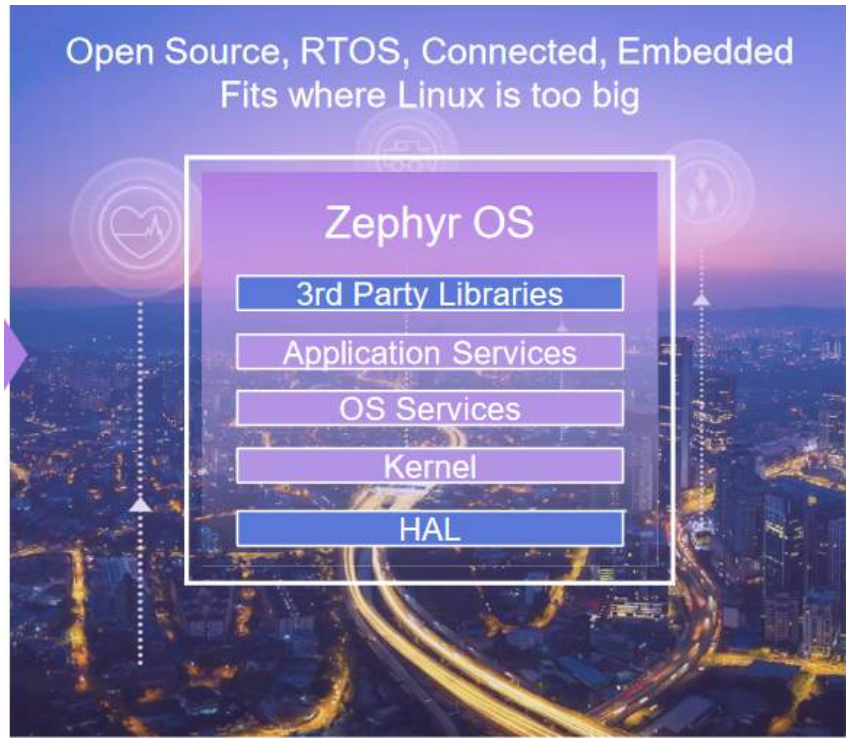
Αρχικά ξεκινάμε με ένα μικρό μοντέλο ,το οποίο ικανοποιεί τους βασικούς στόχους του προβλήματός μας .Στη συνέχεια ελέγχουμε εάν το μοντέλο χωράει στη μνήμη του μικροελεγκτή και μόνο τότε αρχίζουμε να χρησιμοποιούμε πιο σύνθετα μοντέλα στην εκπαίδευση για να βελτιώσουμε την ακρίβεια.Επιπλέον πρέπει να διασφαλιστεί ότι η επεξεργασία των δεδομένων εισόδου κατα την εκπαίδευση σε high level γλώσσα μπορεί να υποστηριχθεί και από την πλευρά του μικροελεγκτή.Αφού αυξάνεται η πολυπλοκότητα των πράξεων στα δεδομένα εισόδου τόσο αυξάνεται και οι χρόνοι υπολογισμών στη συσκευή. Επειδή το μέγεθος του παραγόμενου μοντέλου είναι αρκετά θα το μετατρέψουμε χρησιμοποιώντας το tensorflow lite converter tool .Ωστόσο για εκτελεστεί στον μικροελεγκτή χρειάζεται να είναι σε δυαδική μορφή και για αυτό γίνεται με το command line too xxd .Στο τελικό στάδιο τρέχουμε το μοντέλο και αξιολογούμε την ακρίβειά του πάνω στη συσκευή.

2.3.1.1 Post-Training quantization

Με τη χρήση κβαντισμού προσεγγίζουμε τους αριθμούς κινητής υποδιαστολής με 8-bit ακεραίους .Κατά τον υπολογισμό των νευρωνικών δικτύων μπορούμε να κβαντίσουμε βάρη , σταθερές και ενδιάμεσες τιμές διαφορετικών νευρώνων. Επομένως αλλάζοντας τους 32-bit δεκαδικούς σε 8-bit ακεραίους υπάρχει μείωση στην πολυπλοκότητα των υπολογισμών και κατα συνέπεια στο μέγεθος του μοντέλου και την ταχύτητα εκτέλεσης. Επιπλέον αρκετοί μικροελεγκτές δεν διαθέτουν μονάδα επεξεργασίας floating-point και τα κβαντισμένα μοντέλα είναι η μόνη λύση για αυτούς .

2.4 Zephyr RTOS

Το Zephyr OS είναι ένα ελεύθερο λειτουργικό σύστημα πραγματικού χρόνου (RTOS) ,το οποίο χρησιμοποιείται για τον προγραμματισμό ενσωματωμένων συσκευών διαφορετικών αρχιτεκτονικών [14].Ο πρωταρχικός λόγος δημιουργίας του Zephyr ήταν να αποτελέσει το αντίστοιχο Linux για embedded ,ώστε να υπάρχει μεγαλύτερη ευελιξία στον προγραμματισμό μικροελεγκτών και να μην δεσμευόμαστε από συγκεκριμένες πλατφόρμες .Για αυτό στο zephyr project συνεχώς προστίθενται οδηγοί για περιφερειακά και υποστήριξη για νέες συσκευές .



Εικόνα 2.9*:Structure of Zephyr OS*

Το Zephyr ξεκίνησε ως συνεργασία του Linux Foundation με πρωταρχικά μέλη Intel, NXP Semiconductors, Synopsys, Linaro , Texas Instruments, DeviceTone, Nordic Semiconductor, Oticon και Bose .Στα τέλη Αυγούστου του 2020 είχε τα περισσότερα ενεργά μέλη (contributors) και τον μεγαλύτερο αριθμό commits ,σε σχέση με τα υπόλοιπα RTOS .Το γεγονός ότι υποστηρίζει πρωτόκολλα επικοινωνίας όπως Bluetooth Low Energy,802.15.4,Wifi ,CoAp και MQTT το καθιστά κατάλληλο για εφαρμογές IoT.

Εικόνα 2.10 *:Zephyr Partners [15]*

2.4.1 Configuration

Το Zephyr OS υποστηρίζει πολλές διαφορετικές πλατφόρμες ,οι οποίες διαφέρουν σε μνήμη,περιφερειακά και διευθύνσεις .Επιπλέον το Zephyr διαθέτει λειτουργίες ,που πιθανόν ο προγραμματιστής να μην θέλει να ενσωματώσει στο project λόγω περιορισμένης μνήμης ,ισχύος ή υποπλοκότητας .Έτσι για να υπάρχει αυτή η ευελιξία επιλογών για τους χρήστες ,το zephyr έχει δύο διαφορετικά συστήματα .Το πρώτο είναι τα device trees, τα οποία χρησιμοποιούνται για την περιγραφή του hardware που τρέχει το Zephyr . Το άλλο είναι το Kconfig ,που είναι υπεύθυνο για την πραγματική ρύθμιση του πυρήνα (kernel).

Το device tree δεν ρυθμίζει άμεσα το zephyr ,αλλά το ενημερώνει σχετικά με τα διαθέσιμα περιφερειακά ,τις διευθύνσεις τους ,τα interrupts κ.ο.κ .Το Linux επίσης υποστηρίζει device trees για παρόμοιους σκοπούς.Μια διαφορά στη χρήση των device trees στο Linux και στο Zephyr είναι ότι στο πρώτο μεταγλωττίζει τα device trees με τον device tree compiler (DTC) σε ένα binary blob ,το οποίο φορτώνεται στη μνήμη με τον bootloader.Όταν ο πυρήνας του Linux ξεκινήσει ,θα διαβάσει αυτό το binary blob για να ρυθμίσει σωστά τους drivers .Από την άλλη πλευρά το zephyr χρησιμοποιεί τον DTC για να συνδυάσει όλα τα device tree αρχεία σε ένα νέο device tree αρχείο. Στη συνέχεια αυτό το αρχείο αναλύεται μαζί με ένα σύνολο YALM αρχείων κάποιο python script ,ώστε να παράγει C preprocessor #defines για τους drivers .Παρακάτω επισυνάπτονται μέρος του device tree για το nucleo_f401re και μέρος της εξόδου του που προκύπτει μετά την μεταγλώττιση .

```
leds {
        compatible = "gpio-leds";
        green_led_2: led_2 {
                gpios = <&gpioa 5 GPIO_ACTIVE_HIGH>;
                label = "User LD2";
        };
};
```

Part of device tree for nucleo_f401re (nucleo_f401re.dts)

```
/* Generic property macros: */
#define DT_N_S_leds_S_led_2_P_gpios_IDX_0_EXISTS 1
#define DT_N_S_leds_S_led_2_P_gpios_IDX_0_PH
DT_N_S_soc_S_pin_controller_40020000_S_gpio_40020000
#define DT_N_S_leds_S_led_2_P_gpios_IDX_0_VAL_pin 5
#define DT_N_S_leds_S_led_2_P_gpios_IDX_0_VAL_pin_EXISTS 1
#define DT_N_S_leds_S_led_2_P_gpios_IDX_0_VAL_flags 0
#define DT_N_S_leds_S_led_2_P_gpios_IDX_0_VAL_flags_EXISTS 1
#define DT_N_S_leds_S_led_2_P_gpios_FOREACH_PROP_ELEM(fn) fn(DT_N_S_leds_S_led_2,
gpios, 0)
#define DT_N_S_leds_S_led_2_P_gpios_LEN 1
#define DT_N_S_leds_S_led_2_P_gpios_EXISTS 1
#define DT_N_S_leds_S_led_2_P_label "User LD2"
#define DT_N_S_leds_S_led_2_P_label_FOREACH_PROP_ELEM(fn) fn(DT_N_S_leds_S_led_2,
label, 0) \
   fn(DT_N_S_leds_S_led_2, label, 1) \
   fn(DT_N_S_leds_S_led_2, label, 2) \
   fn(DT_N_S_leds_S_led_2, label, 3) \
   fn(DT_N_S_leds_S_led_2, label, 4) \
   fn(DT_N_S_leds_S_led_2, label, 5) \
   fn(DT_N_S_leds_S_led_2, label, 6) \
   fn(DT_N_S_leds_S_led_2, label, 7)
#define DT_N_S_leds_S_led_2_P_label_EXISTS 1
```

Part of Generated device tree after compilation for nucleo_f401re (devicetree_unfixed.h)

Ο κύριος λόγος της διαφορετική χρήση του παραγόμενου από το DTC binary blob είναι το τεράστιο μέγεθός του , καθώς σε συσκευές με περιορισμένη μνήμη αποτελεί σημαντικό πρόβλημα.

Το Kconfig χρησιμοποιείται για την διαμόρφωση του πυρήνα σε Linux και είναι μια γλώσσα που περιγράφει ρυθμιστικές επιλογές και τη μεταξύ τους σχέση . Σε αντίθεση με τα device trees στο Kconfig πρέπει να προσδιοριστούν οι διαθέσιμες επιλογές ρυθμίσεων ,για να ενεργοποιηθούν τα αντίστοιχα απαραίτητα πακέτα και βιβλιοθήκες . Κάθε εφαρμογή στο zephyr διαθέτει το αρχείο prj.conf ,που περιέχει macros για το configuration του project .

```
CONFIG_PRINTK=y
CONFIG_HEAP_MEM_POOL_SIZE=256
CONFIG_ASSERT=y
CONFIG_GPIO=y
```

Example of a prj.conf file from a zephyr application

Η έξοδος από τα αρχεία ρυθμίσεων είναι το header αρχείο autoconf.h ,το οποίο παράγεται μετά τη διαδικασία της μεταγλώττισης. Παρακάτω φαίνεται μέρος τους autoconf.h από το Blinky παράδειγμα στο zephyr για το nucleo_f401re .

```
#define CONFIG_BOARD "nucleo_f401re"
#define CONFIG_SOC "stm32f401xe"
#define CONFIG_SOC_SERIES "stm32f4"
#define CONFIG_NUM_IRQS 85
#define CONFIG_SYS_CLOCK_HW_CYCLES_PER_SEC 84000000
#define CONFIG_HEAP_MEM_POOL_SIZE 0
#define CONFIG_ROM_START_OFFSET 0x0
#define CONFIG_CORTEX_M_SYSTICK 1
#define CONFIG_CLOCK_CONTROL 1
#define CONFIG_SYS_CLOCK_TICKS_PER_SEC 10000
#define CONFIG_BUILD_OUTPUT_HEX 1
#define CONFIG_FLASH_SIZE 512
#define CONFIG_FLASH_BASE_ADDRESS 0x8000000
#define CONFIG_PINMUX_INIT_PRIORITY 45
#define CONFIG_SERIAL 1
#define CONFIG_CLOCK_CONTROL_STM32_CUBE 1
#define CONFIG_UART_STM32 1
#define CONFIG_GPIO_STM32 1
#define CONFIG_PINMUX_STM32 1
#define CONFIG_ZEPHYR_HAL_NORDIC_MODULE 1
#define CONFIG_ZEPHYR_MBEDTLS_MODULE 1
#define CONFIG_ZEPHYR_SOF_MODULE 1
#define CONFIG_ZEPHYR_TRACERECORDER_MODULE 1
#define CONFIG_ZEPHYR_TRUSTED_FIRMWARE_M_MODULE 1
#define CONFIG_ZEPHYR_NANOPB_MODULE 1
#define CONFIG_ZEPHYR_TENSORFLOW_MODULE 1
#define CONFIG_HAS_CMSIS_CORE 1
#define CONFIG_HAS_CMSIS_CORE_M 1
#define CONFIG_HAS_STM32CUBE 1
#define CONFIG_USE_STM32_LL_UTILS 1
#define CONFIG_BOARD_NUCLEO_F401RE 1
#define CONFIG_SOC_SERIES_STM32F4X 1
#define CONFIG_CPU_HAS_ARM_MPU 1

// ...
```

Parts of the autoconf.h generated when building the “Blinky” sample for the nucleo_f401re

2.4.2 Zephyr’s Toolchain

Η διαδικασία κατασκευής ενός προγράμματος με zephyr ώστε από τον πηγαίο κώδικα να παραχθεί το αντίστοιχο δυαδικό ή δεκαεξαδικό αρχείο για το μικροελεγκτή ,μπορεί χωριστεί σε διάφορα βήματα. Αρχικά στη φάση του configuration χρησιμοποιείται το CMake για να ελέγξει το περιβάλλον και να δημιουργήσει οδηγίες για το “χτίσιμο” του zephyr .Η επόμενη είναι η build φάση όπου είτε το make είτε το ninja χρησιμοποιούνται για να χτιστεί ο πηγαίος κώδικας και μπορεί να χωριστεί σε διάφορα βήματα : generation ,build,link,post-build.Η διαδικασία φαίνεται στο εικόνα 2.11 .

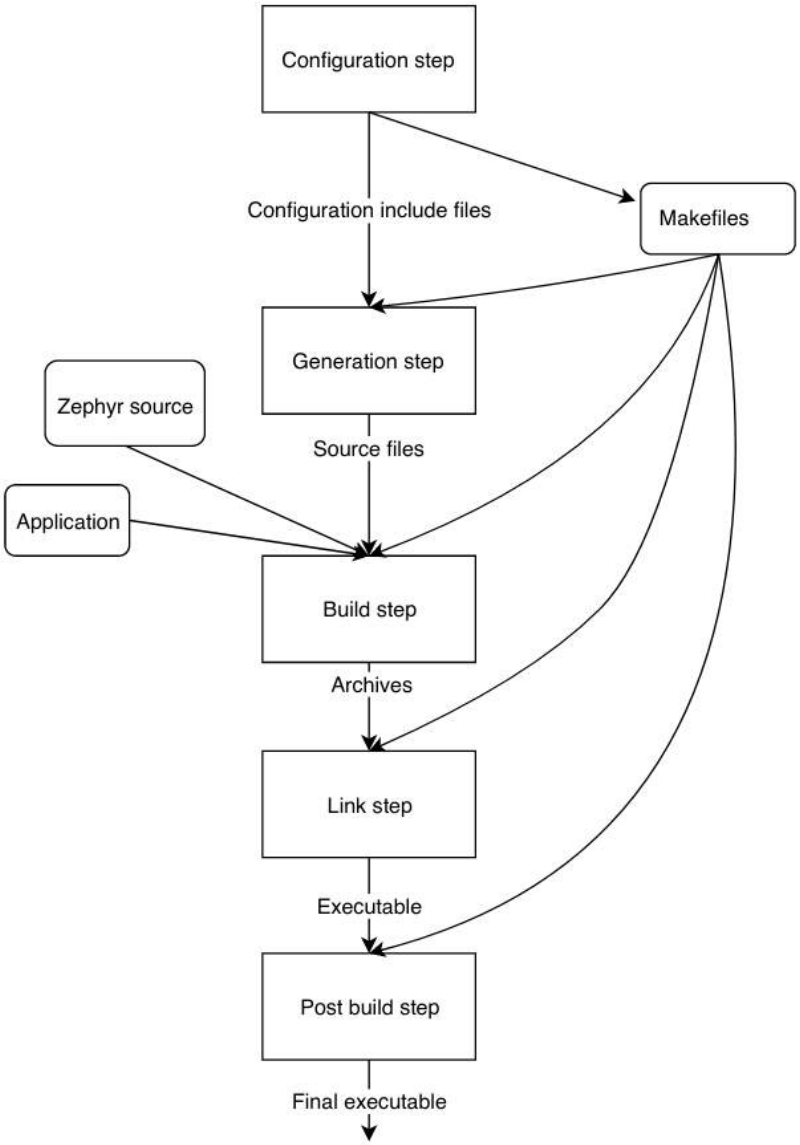
Κατά το generation βήμα το zephyr παράγει περισσότερο C κώδικα , ο οποίος περιλαμβάνει κομμάτια από την υποδομή του για system calls και μερικά αρχεία για την ανίχνευση αντικειμένων πυρήνα .Στο build βήμα το zephyr μεταγλωττιζεται σε μια σειρά στατικά συνδεδεμένων βιβλιοθηκών .Η libzephyr.a και libkernel.a είναι από τις σημαντικότερες βιβλιοθήκες ,καθώς εκεί ορίζεται το μεγαλύτερο μέρος του λειτουργικού συστήματος .Οι υπόλοιπες αντιστοιχούν σε άλλα υποσυστήματα , drivers ή είναι εξαρτώμενες από το υλικό (hardware) .Η εφαρμογή του χρήστη χτίζεται ως μία στατική βιβλιοθήκη με όνομα libapp.a .

Στο στάδιο της διασύνδεσης (link step) , όλες οι βιβλιοθήκες συνδέονται μεταξύ τους.Έτσι η εφαρμογή και τα αντικείμενα του zephyr ενώνονται σε ένα εκτελέσιμο ,το οποίο με τη σειρά του αναλύεται για να εξαχθούν οι ρουτίνες εξυπηρέτησης διακοπών και παράγεται νέο C αρχείο με τον πίνακα των διακοπών .Στη συνέχεια όλες οι βιβλιοθήκες συνδέονται ξανά ,ώστε να παραχθεί το τελικό εκτελέσιμο .

Τέλος κατά το post-build βήμα τρέχει ένα script ,που ελέγχει την ορθότητα του εκτελεσιμου , το μετατρέπει σε κατάλληλη μορφή για να τρέξει στο μικροελεγκτή και βγάζει μηνύματα για την αποσφαλμάτωση .

```
[1/138] Preparing syscall dependency handling

[131/138] Linking C executable zephyr/zephyr_prebuilt.elf

[138/138] Linking C executable zephyr/zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:       14288 B       512 KB      2.73%
            SRAM:        4512 B        96 KB      4.59%
        IDT LIST:          0 GB         2 KB      0.00%
```

Εικόνα 2.11 *:Build process result for blinky example in nucleo_f401re*



Εικόνα 2.12 *:Visualization of Zephyr's build process*

2.4.2 Κλήσεις Συστήματος

Τα νήματα (threads) σε επίπεδο χρήστη δεν έχουν απεριόριστη ελευθερία ,καθώς τρέχουν σε μη-προνομιούχο κατάσταση CPU .Έτσι συγκεκριμένες εντολές και λειτουργίες δεν μπορούν να εκτελεστούν .Επιπλέον δεν έχουν πρόσβαση σε ολόκληρη τη μνήμη και δεν μπορούν να αλλάξουν αντικείμενα του πυρήνα (kernel objects) .

Τα kernel objects διακρίνονται σε τρεις κατηγορίες :

- core kernel object
- thread stack
- device driver instance

Στο πρώτο κατατάσσονται σημαφόροι,νήματα,pipes ,ενώ τα device driver instances είναι δομές (structs), οι οποίες περιέχουν δείκτες σε συναρτήσεις που χρησιμοποιούνται από τον driver.

Επομένως ένα νήμα σε επίπεδο χρήστη για να έχει πρόσβαση στα παραπάνω αντικείμενα ή λειτουργίες χρησιμοποιούνται κλήσεις συστήματος (system calls) .Στον πυρήνα του Zephyr ορίζονται ειδικές συναρτήσεις, οι οποίες μπορεί να κληθούν σε επίπεδο χρήστη και να τρέξουν σε επίπεδο πυρήνα .Επίσης για να μπορούν οι χρήστες να γράφουν πιο εύκολα κλήσεις συστήματος ,το Zephyr παράγει αυτόματα τον κώδικα για την εναλλαγή της κατάστασης του πυρήνα .

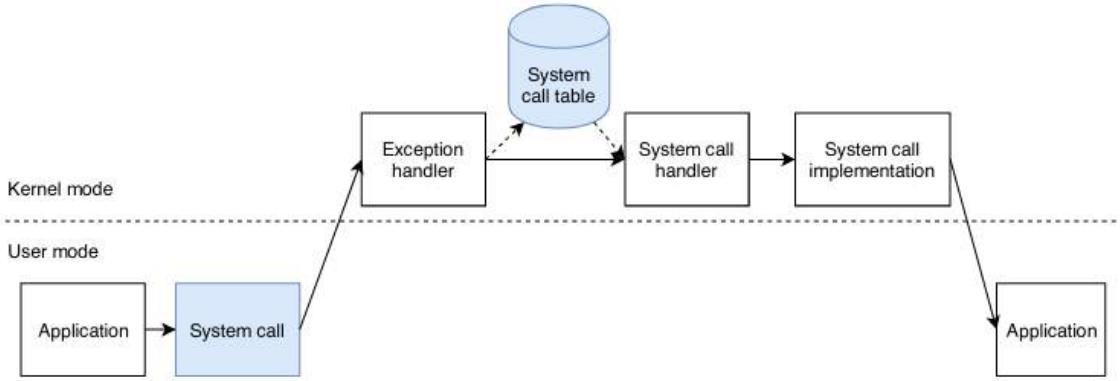
Στο Zephyr κάθε κλήση συστήματος περιλαμβάνει τα τρία παρακάτω στοιχεία :

- C prototype συνάρτηση
- ρουτίνα εξυπηρέτησης (handler)
- συνάρτηση υλοποίησης (implementation function)

Η C prototype διαθέτει το πρόθεμα __syscall για το API του zephyr και δηλώνεται σε κάποιο header αρχείο. Ωστόσο δεν υλοποιείται από τον χρήστη αλλά δημιουργείται αυτόματα από ένα python script [16] .Αυτό το script παράγει κώδικα για την κλιμάκωση δικαιωμάτων του χρήστη ,ελέγχει την ορθότητα των παραμέτρων εισόδου με τον handler και εκτελεί την συνάρτηση υλοποίησης ,οπου και βρίσκεται ο πραγματικός κώδικας της κλήσης συστήματος .

Η ρουτίνα εξυπηρέτησης δηλώνεται χρησιμοποιώντας κάποιο macro που επεκτείνεται στη σωστή δήλωση της συνάρτησης .Μέσα στον handler υπάρχει ένα σύνολο μακροεντολών με τις οποίες επιβεβαιώνονται τα ορίσματα εισόδου .Όταν αυτά είναι σωστά τότε η ρουτίνα εξυπηρέτησης καλεί την συνάρτηση υλοποίησης και επιστρέφει το αποτέλεσμα του system call.

Στο zephyr χρησιμοποιούνται κάποια python scripts ,τα οποία αναζητούν τη __syscall δήλωση της συνάρτησης του system call και αποθηκεύουν τα στοιχεία της σε ένα JSON αρχείο.Στη συνέχεια ένα άλλος python κώδικας δέχεται ως όρισμα το JSON και παράγει C κώδικα και μακροεντολές .



Εικόνα 2.13 *:Control flow of Zephyr's system call dispatch (shadow boxes generated by python scripts)*

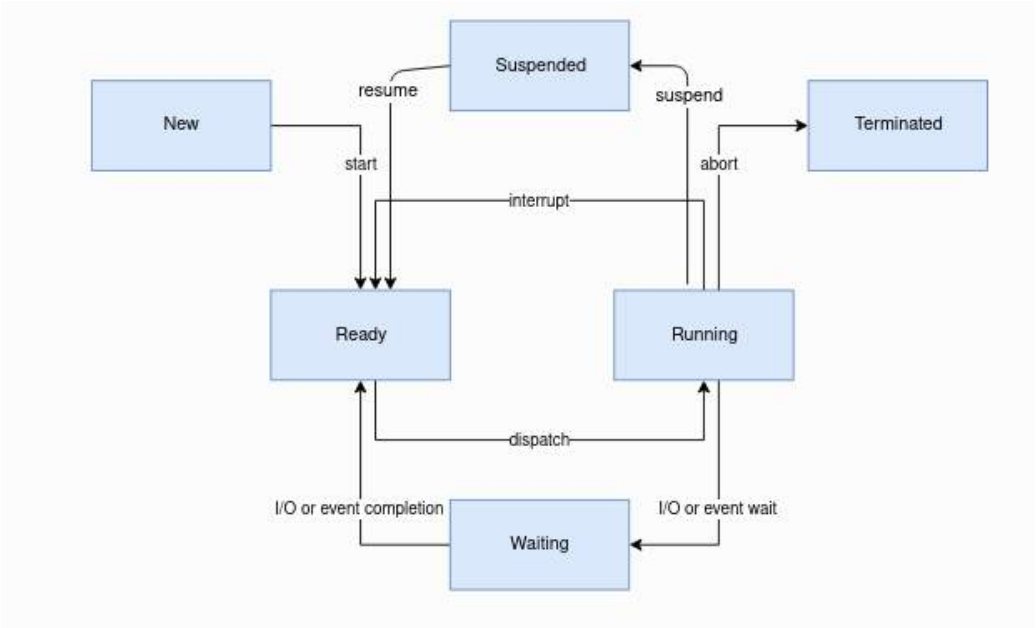
'Οπως φαίνεται στο παραπάνω σχήμα όταν βρισκόμαστε σε επίπεδο χρήστη (user mode) θα πρέπει να μεταβούμε σε επίπεδο πυρήνα (kernel mode) για να εκτελεστεί η κλήση συστήματος .Για να γίνει όμως αυτό πρέπει πρώτα η CPU να μπει σε προνομιούχα (privileged) κατάσταση μέσω exception .

Όμως προτού εκτελεστεί ο κώδικας για τη κλήση συστήματος πρέπει να εξασφαλιστεί ότι δεν θα τρέξει κάτι αυθαίρετο σε privileged κατασταση CPU. Στο Zephyr καταχωρείται περιεχόμενο σε ένα πίνακα ,που περιέχει τις συναρτήσεις της ρουτίνας εξυπηρέτησης.Στη συνέχεια ο exception handler ελέγχει αν αυτό το περιεχόμενο βρίσκεται μέσα στα όρια του πίνακα των system calls .Εάν αληθεύει αυτό τότε αλλάζει τη κατάσταση του επεξεργαστή σε προνομιούχα , ετοιμάζει τη στοίβα και τους καταχωρητές για τη κλήση συστήματος .Τέλος όταν επιστρέψει το system call ,τότε η CPU μεταβαίνει πάλι σε μη-προνομιούχα κατάσταση προτού τελειώσει η συνάρτηση της εφαρμογής.

2.4.3 Χρονοδρομολογητής

Στο zephyr ο χρονοδρομολογητής μοιράζει στις διεργασίες (threads) τον χρόνο εκτέλεσης στον επεξεργαστή με βάση τη μέγιστη προτεραιότητα [17].Η κατάσταση των διεργασιών κατά την εκτέλεση της εφαρμογής αλλάζουν ανάλογα με την ύπαρξη συνθηκών που τις επηρεάζουν όπως interrupts , events ,time slice limit κ.α .

Στο παρακάτω σχήμα φαίνονται οι πιθανές εναλλαγές των καταστάσεων στα threads .

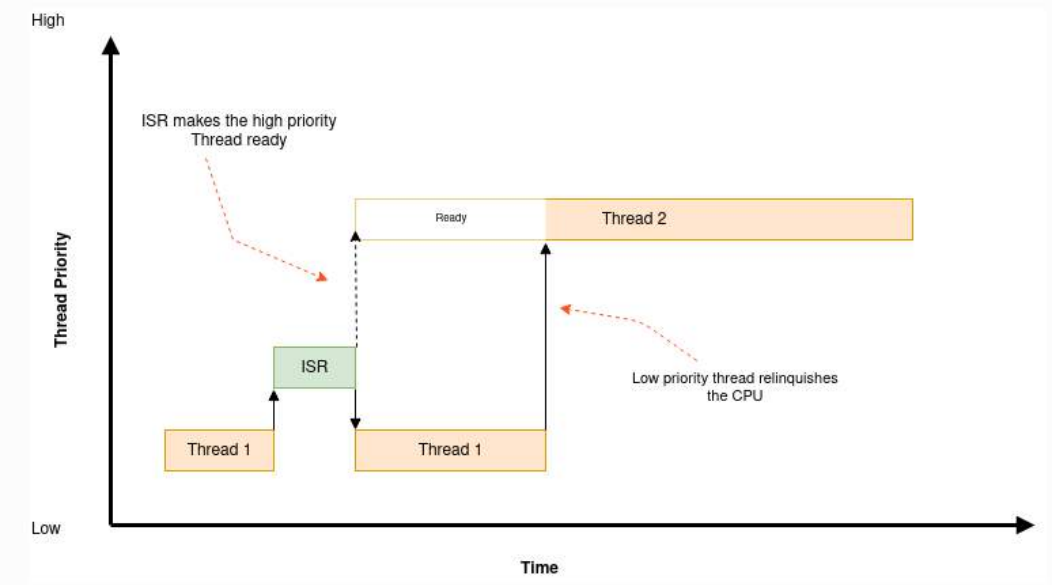


Εικόνα 2.14 *:Change of States in threads*

Οι αλγόριθμοι χρονοδρομολόγησης που μπορεί να χρησιμοποιήσει το zephyr είναι οι εξής :

- Συνεργατικός Διαμοιρασμός Χρόνου (**Cooperative Time Slicing**)
- Διακοπτός Διαμοιρασμός Χρόνου (**Preemptive Time Slicing**)

Κατά την εφαρμογή του πρώτου αλγόριθμου μια συνεργατική διεργασία ξεκινάει να εκτελείται στον επεξεργαστή ,η οποία τον μονοπολεί μέχρι να ολοκληρωθεί .Με αποτέλεσμα στη περίπτωση χρονοβόρων υπολογισμών να προκαλεί καθυστέρηση στην εκτέλεση άλλων διεργασιών μεγαλύτερης ή ίσης προτεραιότητας .

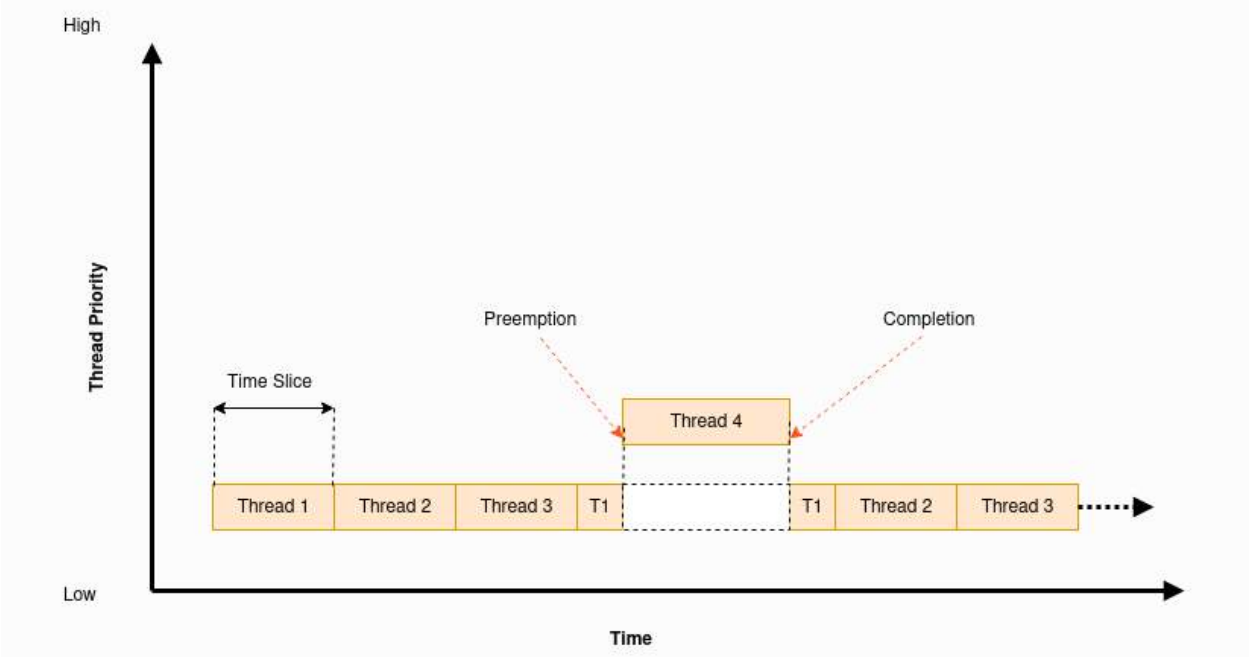


Εικόνα 2.15 :Cooperative Scheduling

Έτσι να να ξεπεραστεί το πρόβλημα της καθυστέρησης κάθε συνεργατική διεργασία μπορεί να απελευθερώσει τον επεξεργαστή για να επιτρέψει σε άλλες να τον χρησιμοποιήσουν .Αυτό επιτυγχάνεται τους δύο παρακάτω τρόπους :

- Με την κλήση της k_yield() ,που βάζει την τρέχων διεργασία στο τέλος της λίστας των διεργασιών προτεραιότητας και ξανακαλεί τον χρονοδρομολογητή
- Με την κλήση της k_sleep() ,η οποία κάνει αναστέλλει το τρέχων thread για κάποιο χρονικό διάστημα και δίνει την ευκαιρία σε άλλες έτοιμες διεργασίες να εκτελεστούν .

Στην περίπτωση της δρομολόγησης με preemptive time slicing η τρέχουσα διεργασία στον επεξεργαστή αλλάζει μόλις εμφανιστεί κάποια με μεγαλύτερη προτεραιότητα.Όπως φαίνεται στο σχήμα 2.16 ο χρονοδρομολογητής διαιρεί τον χρόνο σε slices ,τα οποία μετριούνται σε χτύπους ρολογιού επεξεργαστή . Το μέγεθος των slices ρυθμίζεται και μπορεί να μεταβληθεί κατά την εκτέλεση της εφαρμογής .



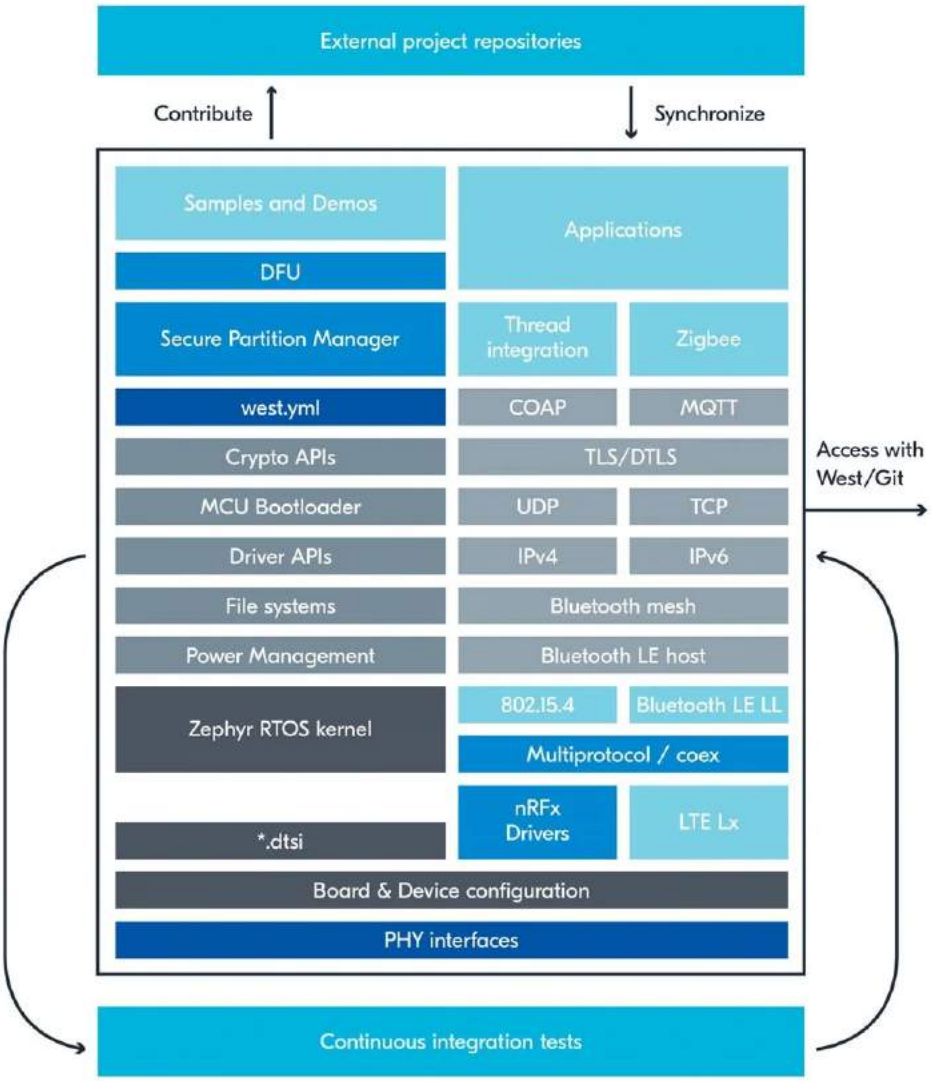
Εικόνα 2.16 :Preemptive Time Slicing

Στο Zephyr RTOS για την χρονοδρομολόγηση των διεργασιών χρησιμοποιείται ένας συνδυασμός preemptive και cooperative ,έτσι όταν υπάρχουν διεργασίες μεγάλης προτεραιότητας time-sensitive τότε επιλέγεται το συνεργατικό scheduling .Ενώ σε περίπτωση χαμηλής προτεραιότητας διεργασιών χρησιμοποιείται το preemptive time slicing .

2.5 nRF Connect sdk

Το nRF Connect SDK είναι ένα open-source software για τη ανάπτυξη εφαρμογών σε επεξεργαστές της Nordic Semiconductors όπως οι nRF52,nRF53 και nRF91 σειρές [18].Επιπλέον παρέχει στους προγραμματιστές ένα εύχρηστο περιβάλλον για την δημιουργία βελτιστοποιημένων σε μέγεθος εφαρμογών για ενσωματωμένες συσκευές περιορισμένης μνήμης .

Έχει ενσωματώσει το Zephyr RTOS μαζί με ένα μεγάλο αριθμό παραδειγμάτων ,βιβλιοθηκών ,πρωτοκόλλων επικοινωνίας και οδηγών όπως φαίνεται στο παρακάτω σχήμα .



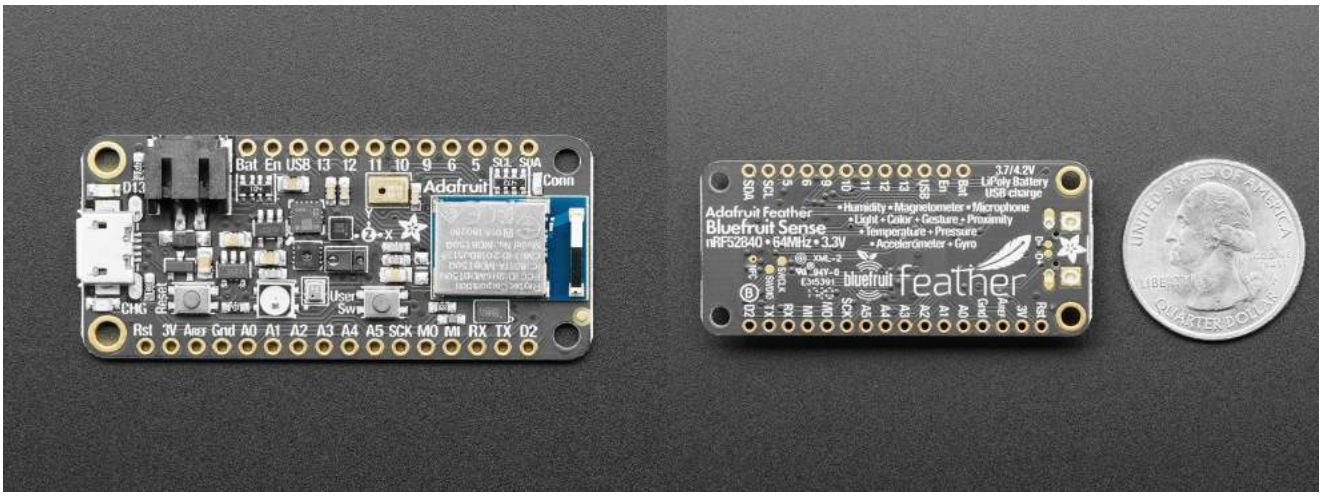
Εικόνα 2.17 *: nRF connect sdk Ecosystem*

Η nordic διαθέτει ενεργό community και παρέχει βοήθεια στους προγραμματιστές μέσω του forum της [19] .Καθώς προσθέτονται νέα στοιχεία ή ενημερώσεις στο nordic sdk connect γίνεται αρκετές παρουσιάσεις για την καλύτερη εξοικείωση των χρηστών με την πλατφόρμα .

Το περιβάλλον του nordic sdk connect μπορεί εύκολα να εγκατασταθεί σε Linux,Windows και macOS , αρκεί κανείς να ακολουθήσει τις οδηγίες που υπάρχουν στο documentation [20] .

2.5.1 Προσθήκη Συσκευής σε nordic SDK

Το adafruit feather sense είναι η συσκευή που επιλέχθηκε για την υλοποίηση της εφαρμογής αναγνώρισης λέξεων καθώς διαθέτει τα απαραίτητα περιφερειακά όπως μικρόφωνο ,bluetooth, αισθητήρες θερμοκρασίας & υγρασίας .Ο nrf52840 επεξεργαστής με FPU (floating point unit) του adafruit feather sense [21] είναι ένας άλλος λόγος για την επιλογή του ,αφού όλη η επεξεργασία των δεδομένων θα γίνεται στη συσκευή .



Εικόνα 2.18 *: Adafruit Feather Sense board*

Τα χαρακτηριστικά είναι τα εξής :

- ARM Cortex M4F (με HW floating point acceleration) στα 64MHz
- 1MB flash και 256KB SRAM
- Bluetooth Low Energy compatible 2.4GHz radio
- 21 GPIO, 6 x 12-bit ADC pins, up to 12 PWM outputs
- Reset button
- SWD debug pads

Τα περιφερειακά του είναι τα παρακάτω :

- PDM microphone : MP34DT01-M [22]
- Temperature and Barometric Pressure sensor: BMP280 [23]
- Gyroscope and Accelerometer : LSM6DS33 [24]
- Humidity sensor: SHT3x-DIS [25]
- Proximity, Light, Gesture, Color sensor: APDS9960 [26]
- Magnetometer: LIS3MDL [27]

Καθώς το feather δεν υποστηρίζεται από το nordic sdk για να χρησιμοποιήσουμε το περιβάλλον και τις δυνατότητές του έπρεπε να το προσθεσουμε χειροκίνητα στο πυρήνα του Zephyr RTOS .Έτσι βάζουμε στο φάκελο με path ncs/zephyr/boards/arm τα κατάλληλα αρχεία που χρειάζονται για τη ρύθμιση του adafruit feather sense .

Παρακάτω φαίνονται τα αρχεία που προστέθηκαν στον φάκελο adafruit_feather_nrf52840_sense μέσα στο directory των boards .

adafruit_feather_nrf52840_sense_defconfig
adafruit_feather_nrf52840_sense.dts
adafruit_feather_nrf52840_sense.yaml
board.cmake
feather_connector.dtsi
Kconfig
Kconfig.board
Kconfig.defconfig

Από αυτά οι σημαντικότερες δηλώσεις για την λειτουργία του επεξεργαστή και των περιφερειακών βρίσκονται στα πρώτα δύο αρχεία .

adafruit_feather_nrf52840_sense.dts

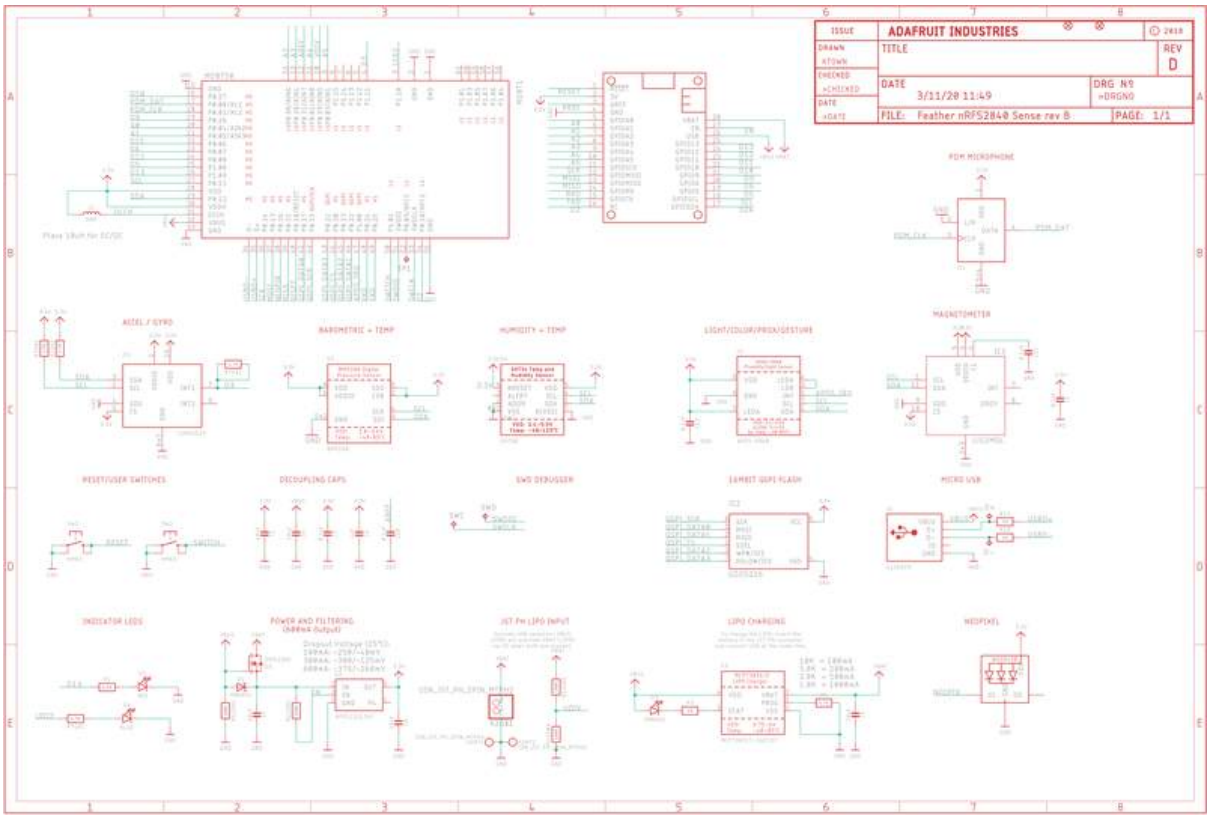
```
/dts-v1/;
#include <nordic/nrf52840_qiaa.dtsi>

/ {
    model = "Adafruit Feather nRF52840 Sense";
    compatible = "adafruit,feather-nrf52840";

    chosen {
            zephyr,console = &uart0;
            zephyr,shell-uart = &uart0;
            zephyr,uart-mcumgr = &uart0;
            zephyr,bt-mon-uart = &uart0;
            zephyr,bt-c2h-uart = &uart0;
            zephyr,sram = &sram0;
            zephyr,flash = &flash0;
            zephyr,code-partition = &slot0_partition;
```

```
    };

    leds {
            compatible = "gpio-leds";
            led0: led_0 {
                    gpios = <&gpio1 9 0>;
                    label = "Red LED";
            };
            led1: led_1 {
                    gpios = <&gpio1 10 0>;
                    label = "Blue LED";
            };
    };

    buttons {
            compatible = "gpio-keys";
            button0: button_0 {
                    gpios = <&gpio1 2 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>;
                    label = "Push button switch";
            };
    };

    /* These aliases are provided for compatibility with samples */
    aliases {
            led0 = &led0;
            led1 = &led1;
            sw0 = &button0;
    };
};

&adc {
    status = "okay";
};

&gpiote {
    status = "okay";
};

&gpio0 {
    status = "okay";
};

&gpio1 {
    status = "okay";
};

&uart0 {
    compatible = "nordic,nrf-uarte";
    current-speed = <115200>;
    status = "okay";
    tx-pin = <25>;
    rx-pin = <24>;
};

&i2c0 {
    compatible = "nordic,nrf-twi";
    status = "okay";
    scl-pin = <11>;
```

```
    sda-pin = <12>;
    clock-frequency = <100000>;
    label = "I2C_0";

    sht3xd@44 {
            compatible = "sensirion,sht3xd";
            reg = <0x44>;
            label = "SHT3XD";
           /*alert-gpios = <&gpio1 10 GPIO_ACTIVE_HIGH>; */
    };

    bme280@77 {
            compatible = "bosch,bme280";
            reg = <0x77>;
            label = "BME280";
    };
};


&spi1 {
    compatible = "nordic,nrf-spi";
    status = "okay";
    sck-pin = <14>;
    mosi-pin = <13>;
    miso-pin = <15>;
};

&qspi {
    status = "okay";
    sck-pin = <19>;
    io-pins = <17>, <22>, <23>, <21>;
    csn-pins = <20>;
    gd25q16: gd25q16@0 {
            /* NOTE: Quad mode not supported as driver does not handle
             * QE bit setting in SR2. Ref. GD25Q16C, Rev 3.0, p. 12.
             */
            compatible = "nordic,qspi-nor";
            reg = <0>;
            writeoc = "pp2o";
            readoc = "read2io";
            sck-frequency = <16000000>;
            label = "GD25Q16";
            jedec-id = [c8 40 15];
            size = <16777216>;
            has-dpd;
            t-enter-dpd = <20000>;
            t-exit-dpd = <20000>;
    };
};

&flash0 {

    partitions {
            compatible = "fixed-partitions";
            #address-cells = <1>;
            #size-cells = <1>;

            boot_partition: partition@0 {
```

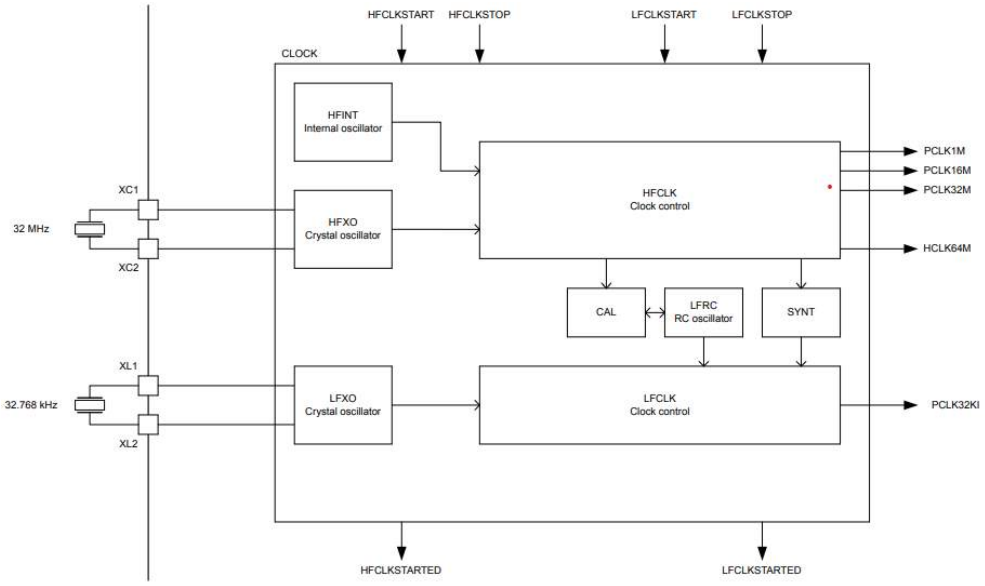
```
                label = "mcuboot";
                reg = <0x000000000 0x00012000>;
        };
        slot0_partition: partition@c000 {
                label = "image-0";
                reg = <0x0000c000 0x00067000>;
        };
        slot1_partition: partition@73000 {
                label = "image-1";
                reg = <0x00073000 0x00067000>;
        };
        scratch_partition: partition@da000 {
                label = "image-scratch";
                reg = <0x000da000 0x0001e000>;
        };

        /*
         * The flash starting at 0x000f8000 and ending at
         * 0x000fffff is reserved for use by the application.
         */

        /* Storage partition will be used by FCB/NFFS/NVS if enabled. */
        storage_partition: partition@f8000 {
                label = "storage";
                reg = <0x000f8000 0x00008000>;
        };
    };
};
```

Οι πληροφορίες για τα πρωτόκολλα επικοινωνίας (uart,i2c,spi) συμπληρώθηκαν με τη βοήθεια του σχηματικού της συσκευής .Επιπλέον επειδή οι οδηγοί για τους αισθητήρες θερμοκρασίας και υγρασίας υπάρχουν ήδη στον πυρήνα του Zephyr RTOS ,μπορούμε απλά να απευθείας να του χρησιμοποιήσουμε αν απλά τους προσθέσουμε με τους σωστούς καταχωρητές τους στο πρωτόκολλο επικοινωνίας τους (κώδικας με κόκκινο) .



Εικόνα 2.19 *: Schematics of Adafruit feather sense [21]*

Για τη χρήση του zephyr RTOS στο adafruit feather sense απαιτείται η ενεργοποίηση του εσωτερικού ταλαντωτή των 32.768 kHz που φαίνεται στο παρακάτω σχήμα .



Εικόνα 2.20 *: Clock Control NRF52840*

adafruit_feather_nrf52840_sense_defconfig

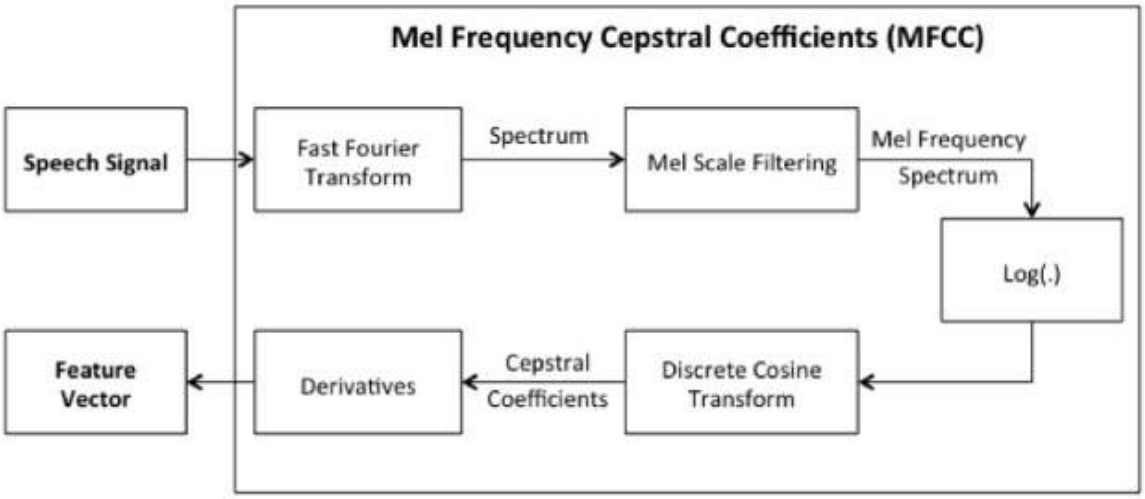
```
CONFIG_SOC_SERIES_NRF52X=y
CONFIG_SOC_NRF52840_QIAA=y
CONFIG_BOARD_ADAFRUIT_FEATHER_NRF52840_SENSE=y

# Enable MPU
CONFIG_ARM_MPU=y

# enable GPIO
CONFIG_GPIO=y

# enable uart driver
CONFIG_SERIAL=y

#enable RTT
CONFIG_USE_SEGGER_RTT=y

# enable console
CONFIG_CONSOLE=y
CONFIG_UART_CONSOLE=y

# additional board options
CONFIG_GPIO_AS_PINRESET=y

# clock config
CONFIG_CLOCK_CONTROL_NRF_K32SRC_RC=y
CONFIG_CLOCK_CONTROL_NRF_K32SRC_20PPM=y
```

Οι παραπάνω μπλε γραμμές ενεργοποιούν τον εσωτερικό ταλαντωτή ο οποίος παρέχει το nrf Real-Time-Counter [28] ,που χρησιμοποιεί το zephyr ως ρολόι του κεντρικού του συστήματός .Έτσι δίχως τη συγκεκριμένη δήλωση δεν πρόκειται να τρέξει το λειτουργικό του zephyr.

2.6 Mel Frequency Cepstral Coefficients (MFCC)

Αποτελεί την πιο γνωστή μέθοδο εξαγωγής δεδομένων στην αυτόματη αναγνώριση φωνής.Η μέθοδος αναπτύχθηκε από τον Mermelstein το 1976 και βασίζεται σε πειράματα πάνω στην ανθρώπινη ερμηνεία λέξεων [29]. Ο αλγόριθμος MFCC προσπαθεί να μιμηθεί κομμάτια του τρόπου που ο άνθρωπος παράγει και αντιλαμβάνεται την ομιλία.Το τελικό διάνυσμα περιλαμβάνει και μεταβολές στο χρόνο για να αναπαραστήσει την δυναμική φύση της ομιλίας .

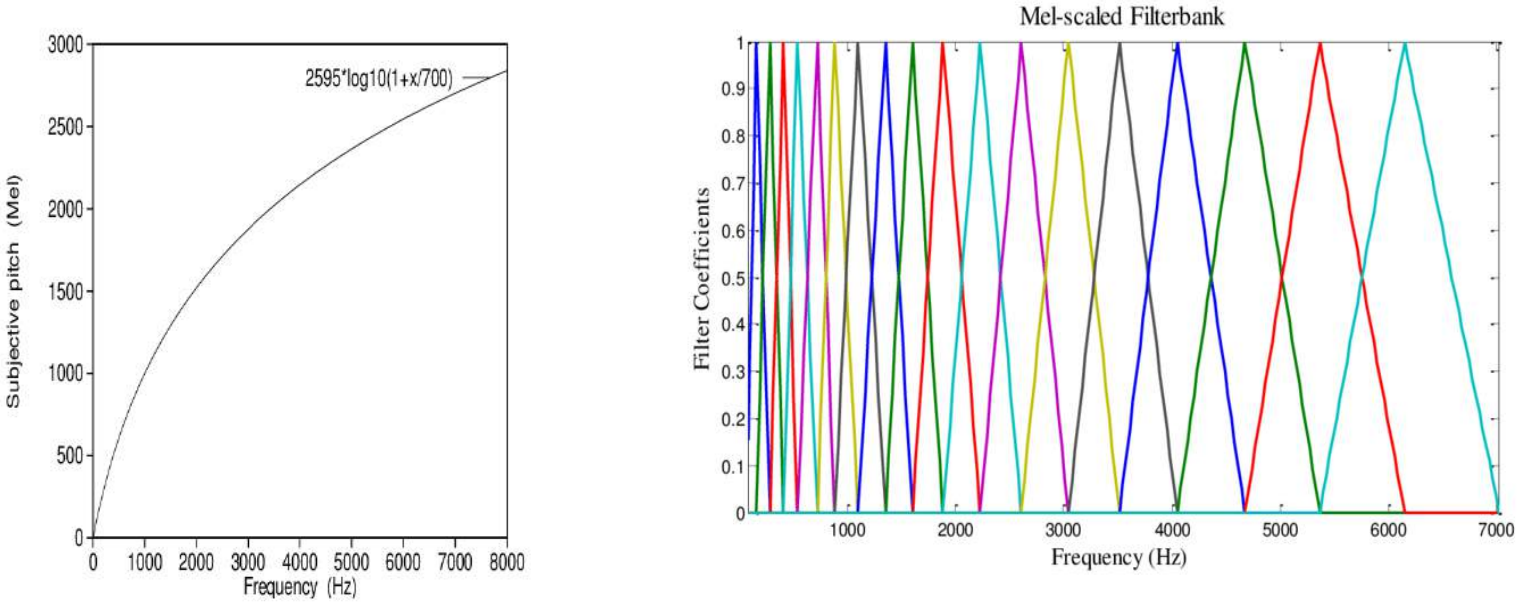
Στο μπλοκ διάγραμμα του σχήματος 2.21 φαίνονται τα βήματα της διαδικασίας υπολογισμού του τελικού διανύσματος.



Εικόνα 2.21 *: MFCC Block Diagram*

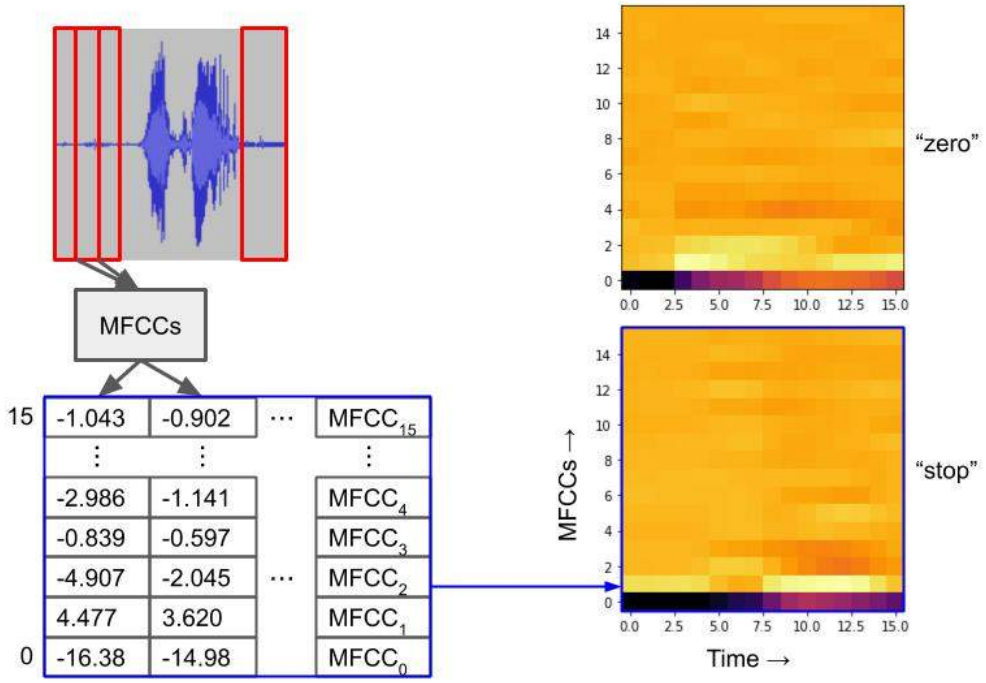
Αρχικά υπολογίζεται ο Διακριτός Μετασχηματισμός Fourier του σήματος εισόδου ώστε να πάρουμε το φάσμα συχνότητας για να υπολογίσουμε το περιοδόγραμμα φάσματος ισχύος.Στη συνέχεια υπολογίζεται το φάσμα των συχνοτήτων Mel φιλτράροντας το περιοδόγραμμα με μία filterbank που αποτελείται από πολλαπλά ζωνοπερατά φίλτρα (σχήμα 2.22) και υπολογίζοντας τις εντάσεις των συχνοτήτων αυτές. Ο αριθμός, το σχήμα και οι κεντρικές συχνότητες των φίλτρων ποικίλουν ανάλογα την εφαρμογή.

Η κλιμακα Mel είναι μία γραμμική κλίμακα και επιλέγεται γιατί μπορεί να προσαρμοστεί στη μη γραμμική τονική αντίληψη του ανθρώπινου ακουστικού συστήματος.



Εικόνα 2.22 *: Mel frequency scale & Filterbank*

Το επόμενο βήμα είναι ο υπολογισμός του λογαρίθμου των παραπάνω εντάσεων.Αυτό βασίζεται σε πειράματα που δείχνουν ότι η ανθρώπινη αντίληψη της έντασης τους ήχου είναι σε λογαριθμική κλίμακα.Ακολούθως υπολογίζουμε τους Cepstral συντελεστές του σήματος βρίσκοντας τον διακριτό μετασχηματισμό συνημιτόνου (DCT) και κρατώντας τις χαμηλές έντασης συχνότητες του cepstrum .Τέλος για να συμπεριληφθεί η χρονική μεταβολή των συντελεστών , το τελικό διάνυσμα επεκτείνεται και με της πρώτης και δεύτερης τάξης παραγώγους των αρχικών συντελεστών .



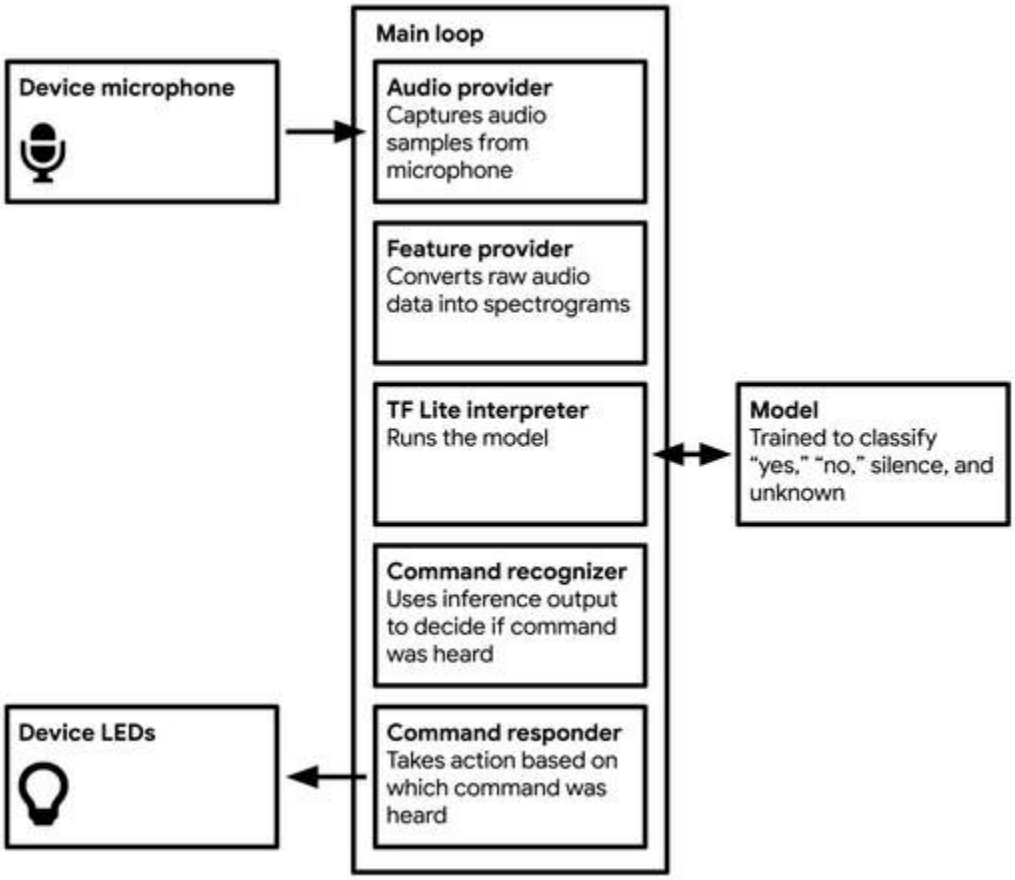
Εικόνα 2.23 *: MFCC Spectrogram*

Οι mfcc συντελεστές μπορούν να παρομοιαστουν με τα pixels μια εικόνας ,καθώς όπως φαίνεται στο παραπάνω σχήμα κάθε παράθυρο αντιστοιχεί σε μια κάθετη στήλη συντελεστών .Έτσι ο πίνακας με τους mfcc συντελεστές αντιστοιχίζεται σε μία λέξη και αυτή την πληροφορία θα δούμε στη συνέχεια ότι θα δοθεί ως είσοδο στον νευρωνικό .

3. Ανάλυση Εφαρμογής Αναγνώρισης Λέξεων

3.1 Tensorflow Microspeech

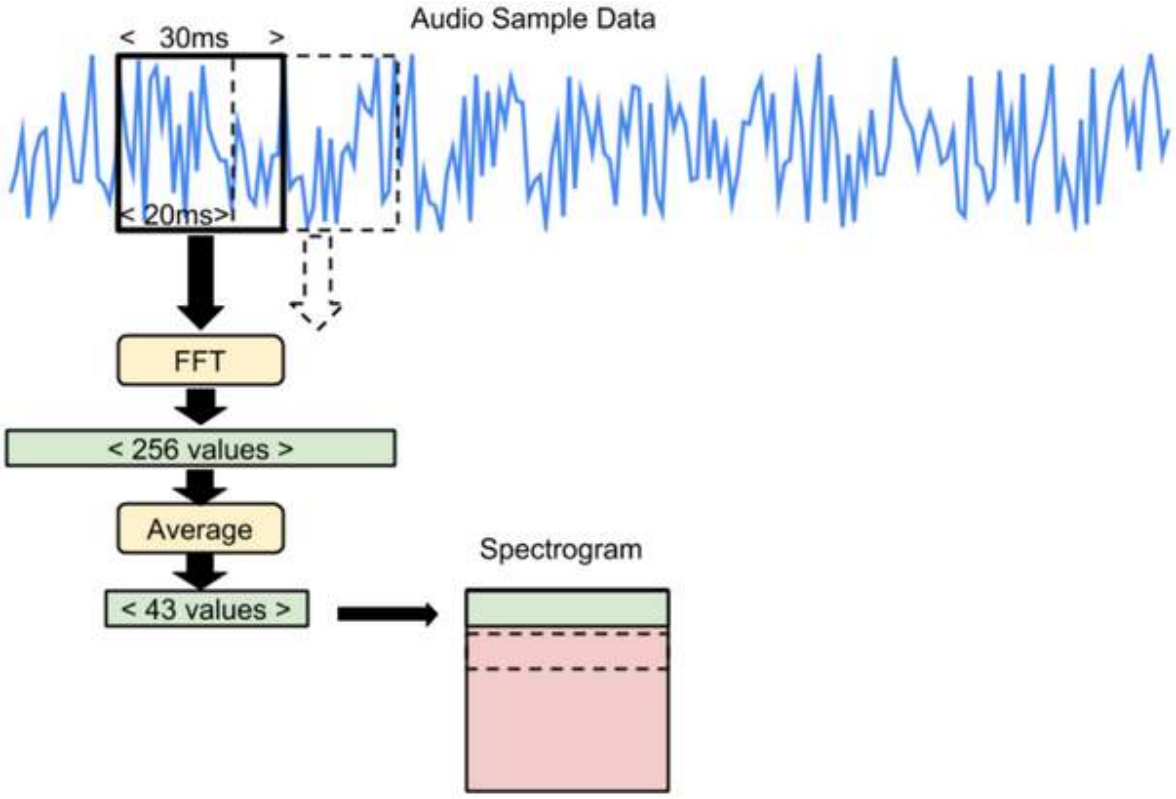
Αρχικά μελετήθηκε το παράδειγμα microspeech της Tensorflow για αναγνώριση λέξεων και προσαρμόστηκε ώστε να ενσωματωθεί στο adafruit feather sense.



Εικόνα 3.1 *: Pipeline of Microspeech Example [8]*

Στο παραπάνω σχήμα φαίνεται η ροή της πληροφορίας του ήχου καθώς περνάει από διάφορα στάδια επεξεργασίας για να γίνει η αναγνώριση συγκεκριμένων λέξεων .

Στο πρώτο κομμάτι του main loop για το audio_provider χρειάστηκε να προστεθεί κώδικας για να γίνεται η δειγματοληψία του ήχου μέσω του pdm μικροφώνου του μικροελεγκτή . Το στάδιο εξαγωγής χαρακτηριστικών (feature provider) μετατρέπει τα δείγματα ήχου σε spectrogram ,δηλαδή προετοιμάζει την είσοδο για τον tflite interpreter.



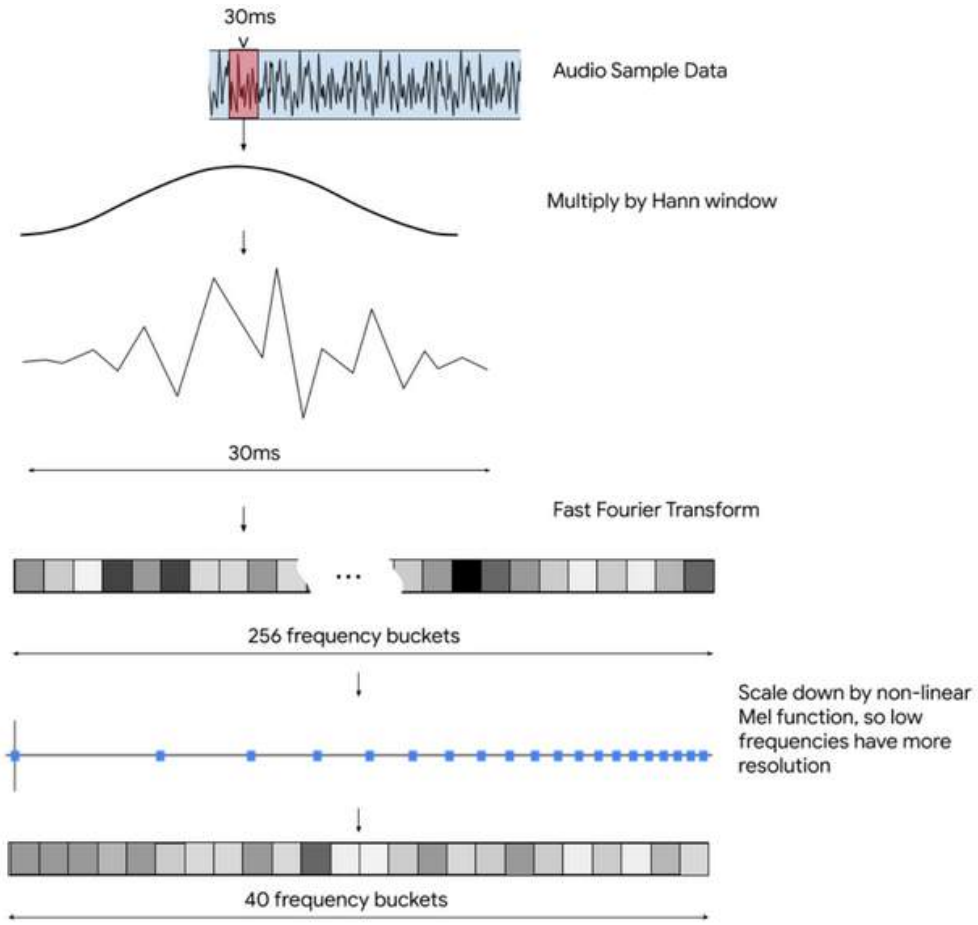
Εικόνα 3.2 *: Feature extraction process*

Στη συνέχεια δημιουργείται ο tflite interpreter από το προεκπαιδευμένο νευρωνικό δίκτυο και του δίνουμε ως είσοδο το spectrogram .Με την έξοδο του inference ερμηνεύεται ποια λέξη ειπώθηκε και ανάλογα με την λέξη μπορούμε να εκτελέσουμε διαφορετικές ενέργειες χρησιμοποιώντας τα περιφερειακά του μικροελεγκτή.

3.1.1 Διαδικασία Εξαγωγής Χαρακτηριστικών

Σε αυτό το σημείο θα παρουσιαστεί πιο αναλυτικά το κομμάτι της επεξεργασίας που χρησιμοποιήθηκε για το feature extraction .Αρχικά ένα παράθυρο 30ms ήχου υπολογίζεται ο Fast Fourier Transform για ένα παράθυρο 30ms ήχου φιλτράρεται από το hann παράθυρο ,για να περιορίσει την επιρροή των άκρων του παραθύρου (smoothing edges) .Στη συνέχεια εφαρμόζεται Fast Fourier Transform (FFT) σε αυτό το παράθυρο των 30 ms . Παρόλο που ο FFT παράγει μιγαδικούς αριθμούς με φανταστικό και πραγματικό μέρος ,εμεις ενδιαφερόμαστε μόνο για την συνολική ενέργεια που

αντιστοιχεί σε κάθε τμήμα (bucket) συχνότητας . Ο Fourier σε N δείγματα παράγει πληροφορία για N/2 συχνότητες , οπότε για 30ms με συχνότητα 16kHz (samples/ sec) χρειάζονται 480 δείγματα .Όμως ο αλγόριθμος FFT χρειάζεται είσοδο σε δύναμη του δύο και για αυτό προσθέτουμε μηδενικά μέχρι να φτάσουμε 512 δείγματα ,τα οποία αντιστοιχούν σε 256 συχνότητες . Στη συνέχεια συρρικνώνουμε τον αριθμό των συχνοτήτων παίρνοντας τη μέση τιμή γειτονικών τιμών σε 40 υποσυχνότητες .Αυτή η υποδειγματοληψία δεν γίνεται γραμμικά αλλά χρησιμοποιεί την κλίμακα συχνοτήτων mel , που βασίζεται στην ανθρώπινη αντίληψη των διαφορετικών συχνοτήτων .Το παράθυρο μετακινείται με βήμα 20ms κάθε φορά και υπάρχει μια μικρή επικάλυψη με το αμέσως επόμενο παράθυρο ,ώστε να αυξάνεται ο συνολικός αριθμός των εισόδων στο μοντέλο με αποτέλεσμα να μεγιστοποιείται η πιθανότητα εύρεσης των λέξεων .

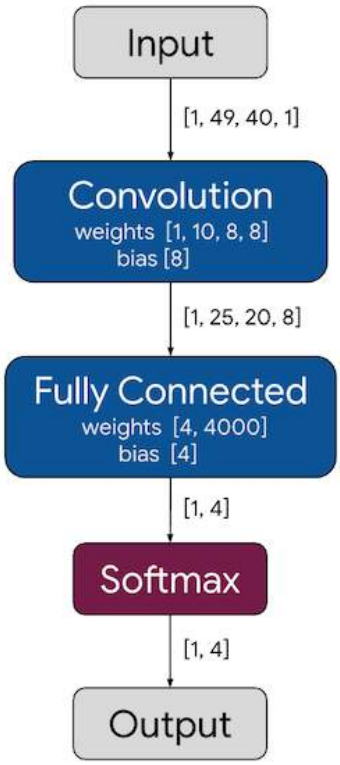


Εικόνα 3.3 *: Audio input processing flow*

3.1.2 Αρχιτεκτονική Νευρωνικού

Πρόκειται για ένα απλό μοντέλο που αποτελείται από :

- Convolutional 2D layer
- Fully Connected Layer
- Softmax layer



Εικόνα 3.4 *: Model Architecture*

3.1.3 Dataset

Για την εκπαίδευση του νευρωνικού χρησιμοποιήθηκε το Google Speech Commands Dataset ,το οποίο αποτελείται από περισσότερα από 105.000 δείγματα ήχων για 30 διαφορετικές λέξεις σε συχνότητα δειγματοληψίας 16kHz .Στον παρακάτω πίνακα παρουσιάζονται οι λέξεις που έχουμε στην διάθεσή μας από το dataset .

Word	Number of Utterances
Backward	1,664
Bed	2,014
Bird	2,064
Cat	2,031
Dog	2,128
Down	3,917
Eight	3,787
Five	4,052
Follow	1,579
Forward	1,557
Four	3,728
Go	3,880
Happy	2,054
House	2,113
Learn	1,575
Left	3,801
Marvin	2,100
Nine	3,934
No	3,941
Off	3,745
On	3,845
One	3,890
Right	3,778
Seven	3,998
Sheila	2,022
Six	3,860
Stop	3,872
Three	3,727
Tree	1,759
Two	3,880
Up	3,723
Visual	1,592
Wow	2,123
Yes	4,044
Zero	4,052

Πίνακας Συσχέτισης λέξεων με αριθμό δειγμάτων

3.1.4 Εκπαίδευση & Εξαγωγή inference

Το Tensorflow παρέχει ένα jupyter notebook [30] για την εκπαίδευση του νευρωνικού καθώς και την εξαγωγή του inference σε μορφή κατάλληλη για να φορτωθεί στον μικροελεγκτή. Για να κατανοήσουμε πως εξάγεται το inference πρέπει πρώτα να αναφέρουμε ότι στο tensorflow κάθε μοντέλο περιλαμβάνει δύο κεντρικά στοιχεία:

- Τα βάρη και τις σταθερές από την εκπαίδευση

- Ένα γράφο με όλες τις διαδικασίες που συνδυάζουν την είσοδο με τα βάροι και τις σταθερές για να παραχθεί η έξοδος του μοντέλου

Έτσι τα δύο παραπάνω στοιχεία πρέπει να ενωθούν σε ένα αρχείο με συγκεκριμένο format (file.pb) ,ώστε να μπορούμε να τρέξουμε inference . Η διαδικασία αυτής της δημιουργίας αρχείου του μοντέλου λέγεται “πάγωμα” (freezing) και αποτελεί μια στατική αναπαράσταση του γράφου μαζί με τα αποθηκευμένα βάρη του μοντέλου .Προφανώς το στιγμιότυπο για το οποίο κρατάμε τα βάρη και το γράφο είναι όταν παρατηρείται μέγιστη ακρίβεια κατά την εκπαίδευση .

Στη συνέχεια χρησιμοποιείται ο tflite-Converter ο οποίος μετατρέπει το προεκπαιδευμένο μοντέλο (file.pb) στα ακόλουθα βελτιστοποιημένα tfilte μοντέλα :

- Float16
- Dynamic
- Full integer

Το τελευταίο αποτελεί την ιδανική επιλογή για μοντέλα σε μικροελεγκτές καθώς ελαχιστοποιεί το χρόνο εκτέλεσης των υπολογισμών.

Τέλος τα νέα tflite μοντέλα πρέπει να μετατραπούν σε μορφή κατάλληλη για το C++ TFlite API ,που τρέχει ο μικροελεγκτής .Αυτό γίνεται με το xxd , ένα linux command line tool ,το οποίο παράγει hex dump από οποιαδήποτε είσοδο πάρει .

3.1.5 Υλοποίηση microspeech σε NRF-connect SDK

Για την διαδικασία του porting του παραπάνω παραδείγματος στο adafruit feather sense χρησιμοποιήθηκε ένα συγκεκριμένο tensorflow branch [31] .Στο nrf-connect sdk για να χτιστεί το project μέσω του zephyr RTOS χρειάζεται μία εξωτερική βιβλιοθήκη (**libtensorflow-microlite.a)** , η οποία περιλαμβάνει κάποιες συναρτήσεις που χρησιμοποιούνται στον κώδικα για εξαγωγή χαρακτηριστικών .

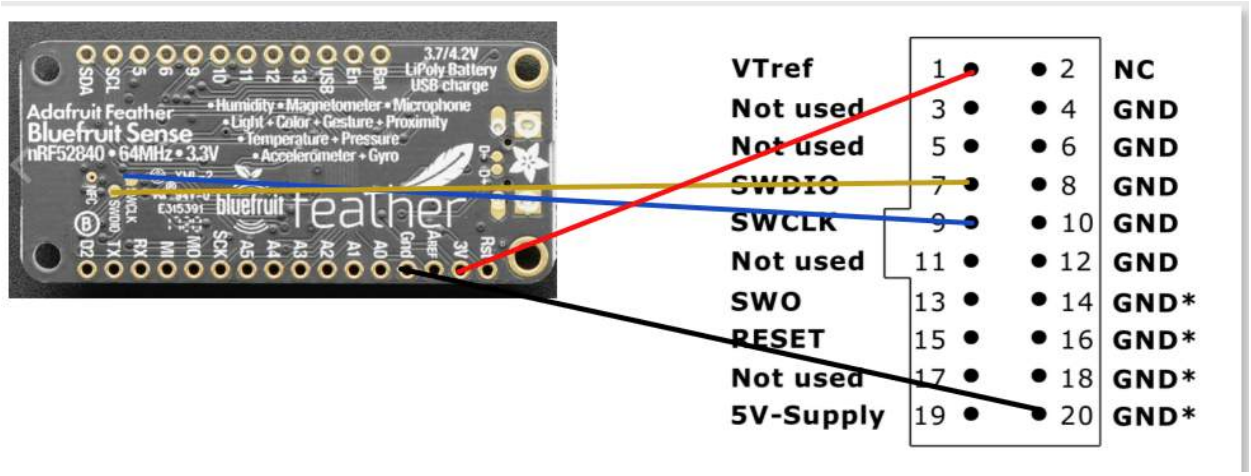
```
make -f tensorflow/lite/micro/tools/make/Makefile
TARGET=adafruit_feather_nr52840_sense  TARGET_ARCH=cortex-m4
generate_micro_speech_make_project
```

Η παραπάνω εντολή εκτελείται μέσω του CMakeLists.txt προτού γίνει η μεταγλώττιση του project με το west (build tool για zephyr) .

```
west build -b adafruit_feather_nrf52840_sense
```

Με την επιτυχημένη εκτέλεση της τελευταίας εντολής παράγεται το δεκαεξαδικό αρχείο της εφαρμογής , το οποίο στη συνέχεια φορτώνεται στο nrf52840 με το εργαλείο **nrfjprog** (Nordic command line flash tool) μέσω του Segger Jlink debugger με SWD .

```
nrfjprog -f nrf52 --program zephyr.hex --sectorerase
```



Εικόνα 3.5 *: Connections of nrf52840 & Jlink for flashing hex file*

Παρακάτω φαίνονται τα αποτελέσματα του microspeech παραδείγματος για ένα προεκπαιδευμένο μοντέλο δύο λέξεων .

```
HEARD unknown (108) @1984ms
HEARD unknown (148) @2944ms
HEARD unknown (157) @3936ms
HEARD unknown (149) @4896ms
HEARD unknown (167) @5888ms
HEARD unknown (173) @6880ms
HEARD unknown (106) @7840ms
HEARD nine (117) @8832ms
HEARD unknown (111) @9824ms
HEARD unknown (165) @10784ms
HEARD unknown (198) @11776ms
HEARD unknown (140) @12736ms
HEARD unknown (123) @13728ms
HEARD unknown (113) @14720ms
HEARD unknown (173) @15680ms
HEARD unknown (137) @16672ms
HEARD visual (108) @17664ms
HEARD visual (127) @18624ms
HEARD unknown (129) @19616ms
HEARD unknown (166) @20608ms
HEARD unknown (179) @21568ms
HEARD unknown (193) @22560ms
HEARD unknown (208) @23520ms
HEARD unknown (143) @24512ms
HEARD nine (119) @26464ms
HEARD nine (119) @27456ms
HEARD visual (158) @28448ms
```

Εικόνα 3.6 *: Uart Output from keyword detection*

Στην εικόνα 3.6 η έξοδος προκύπτει με την κλήση της error_reporter->Report() και όπως βλέπουμε εμφανίζεται κάθε φορά η κλάση αναγνώρισης μαζί το score και το χρόνο μέχρι το τελευταίο reset στο μικροελεγκτή .Αξίζει να αναφερθεί ότι για να δημιουργήσουμε ένα αξιόπιστο μοντέλο αναγνώρισης λέξεων πρέπει να το εκπαιδευσουμε και σε άλλες καταστάσεις όπως στο θόρυβο,ομιλία δηλαδή διαφορετικές λέξεις .Καθώς εάν έχει εκπαιδευτεί μόνο για συγκεκριμένες λέξεις ,όταν λάβει ως είσοδο κάτι άγνωστο θα αναγκαστεί να το βάλει λανθασμένα σε κάποια κλάση .

Ωστόσο δίνεται η δυνατότητα να οριστούν ατομικά thresholds για όλες τις λέξεις ,ώστε να βελτιωθεί η ακρίβεια του μοντέλου .Επιπλέον δεν είναι υποχρεωτικό να βρεθεί ολόκληρη η λέξη ώστε να εκτελεστεί κάποια ενεργεια στο μικροελεγκτή ,αλλά μπορούμε να εντοπίσουμε συγκεκριμένους χαρακτήρες της λέξης όπως φαίνεται στον παρακάτω κώδικα .

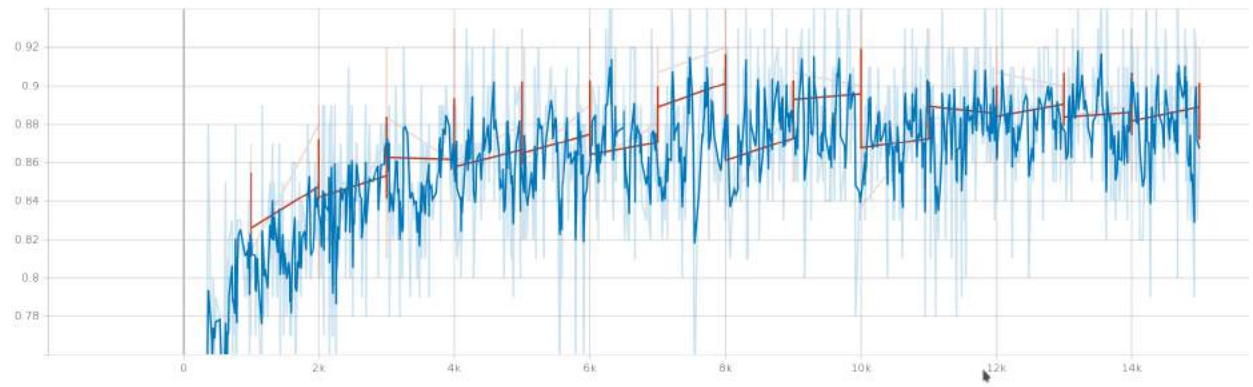
```
#include <drivers/gpio.h>
#include "command_responder.h"
#include <zephyr.h>
#include <string.h>

// ---- defines LED0 for the LED pin ------------------------------------------
// the devicetree node identifier for the "led0" alias.
#define LED0_NODE DT_ALIAS(led0)

#if DT_NODE_HAS_STATUS(LED0_NODE, okay)
#define LED0    DT_GPIO_LABEL(LED0_NODE, gpios)
#define PIN     DT_GPIO_PIN(LED0_NODE, gpios)
#define FLAGS   DT_GPIO_FLAGS(LED0_NODE, gpios)
#endif


#define LED1_NODE DT_ALIAS(led1)

#if DT_NODE_HAS_STATUS(LED1_NODE, okay)
#define LED1    DT_GPIO_LABEL(LED1_NODE, gpios)
#define PIN1    DT_GPIO_PIN(LED1_NODE, gpios)
#endif

// LED cycle defs
#define LED0_CYCLE_TIME 1000
#define LED0_ON_TIME     100
#define LED0_OFF_TIME   LED0_CYCLE_TIME - LED0_ON_TIME

void RespondToCommand(tflite::ErrorReporter *error_reporter,
                      int32_t current_time, const char *found_command,
                      uint8_t score, bool is_new_command)
{


    const struct device *dev,*dev1;
    int rc;
    dev = device_get_binding(LED0);
```

```
    dev1 = device_get_binding(LED1);


    rc = gpio_pin_configure(dev, PIN, GPIO_OUTPUT_ACTIVE | FLAGS);
    gpio_pin_configure(dev1, PIN1, GPIO_OUTPUT_ACTIVE | FLAGS);

    gpio_pin_set(dev, PIN1, (int)true);
    gpio_pin_set(dev, PIN, (int)true);

    static int32_t last_command_time = 0;

  if (is_new_command)


  {
    //TF_LITE_REPORT_ERROR(error_reporter, "HEARD %s (%d) @%dms", found_command,score,
current_time);



   if ((found_command[0]=='n' )&& (found_command[2]=='n')){
            printk("nine found \n");
            gpio_pin_set(dev, PIN, (int)false);
            k_msleep(200);


            }
   else if ((found_command[0]=='v') && (found_command[3]=='u')  &&(found_command[5]=='l')){
            printk("visual found \n");
            gpio_pin_set(dev, PIN1, (int)false);
            k_msleep(200);


            }
   else {
            printk("unknown / noise \n");
            }
   }
}
```

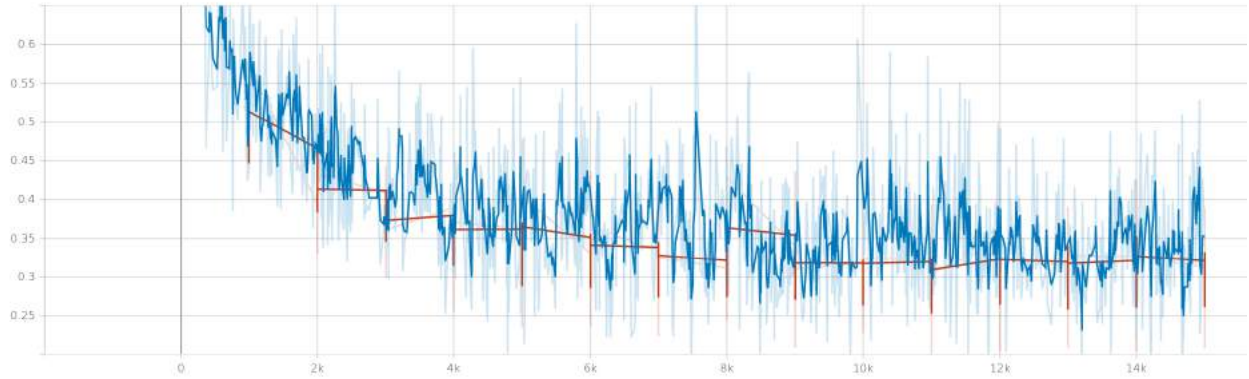
Στο κώδικα παραπάνω ορίζουμε την συνάρτηση RespondToCommnad(),όπου εάν αναγνωριστεί οποιαδήποτε από αυτές τις λέξεις (nine,visual) αναβοσβήνουν τα led του adafruit feather sense .

Στη συνέχεια φαίνονται τα χαρακτηριστικά του προεκπαιδευμένου μοντέλου δύο λέξεων που προέκυψε από το nrf-connect-sdk

words	quantized model accuracy	flash	sram
nine , visual	92,1 %	234 kB	77.5 kB



Εικόνα 3.7 *: Accuracy / epochs diagram with smoothing*



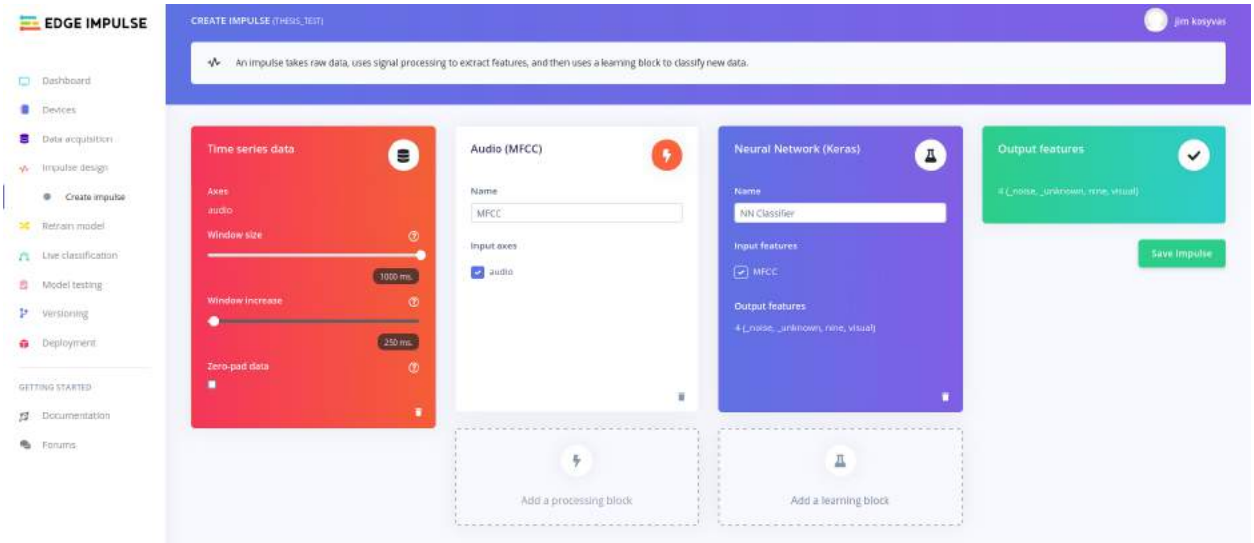
Εικόνα 3.8 *:Loss/ epochs diagram*

Στα δύο παραπάνω σχήματα φαίνεται η μεταβολή της ακρίβειας και των απωλειών καθώς αυξάνεται ο αριθμός των επαναλήψεων .Το μοντέλο στις πρώτες 12000 επαναλήψεις εκπαιδεύεται με learning rate 0.001 και στις τελευταίες 3000 με 0.0001. Αυτό συμβαίνει ώστε να συγκλίνει το νευρωνικό πιο γρήγορα στην αρχή και στο τέλος θέτοντας μικρότερο learning rate να ρυθμιστούν με μεγαλύτερη ακρίβεια τα βάρη και οι σταθερές .

3.2 Σχεδιασμός Νευρωνικού Δικτύου στο Edge Impulse Studio

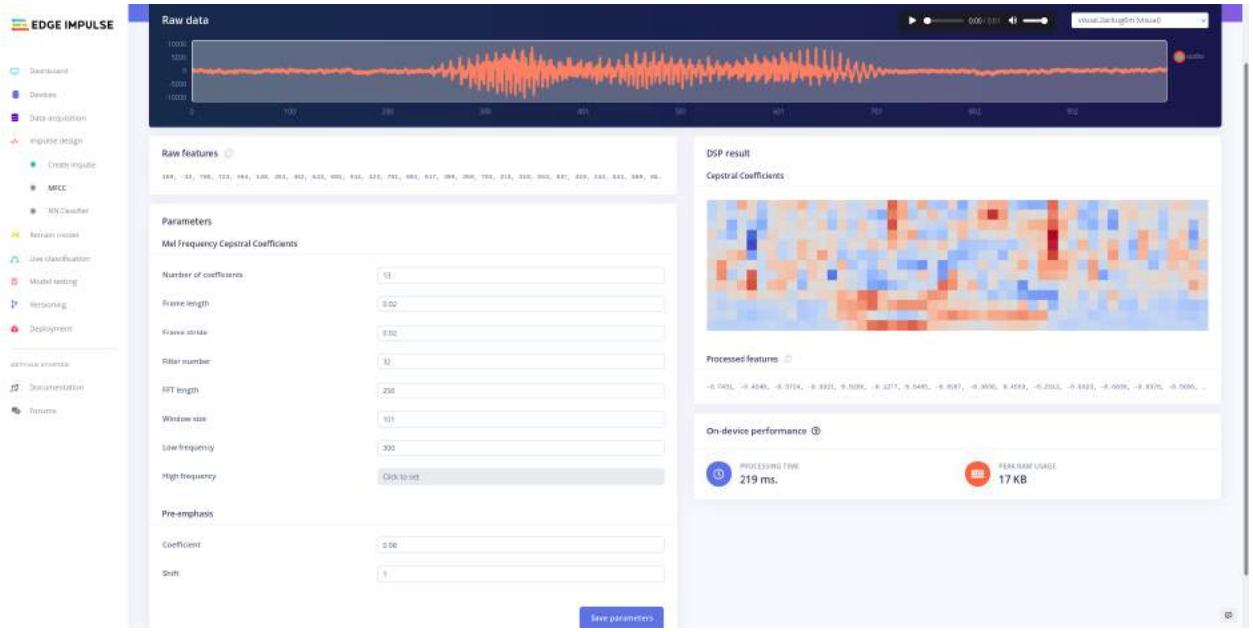
Η διαδικασία δημιουργίας νευρωνικού μοντέλου με χρήση του Edge Impulse είναι αρκετά πιο απλή σε σχέση με την προηγούμενη μέθοδο καθώς πολλά βήματα είναι αυτοματοποιημένα .Αρχικά πρέπει να ανεβάσουμε τα δεδομένα ήχου για την εκπαίδευση στη cloud πλατφόρμα του edge impulse ,είτε μέσω command line tool είτε μέσω του API .Για αυτό το λόγο χρησιμοποιήθηκε το jupyter notebook curation_script [32],το οποίο προσθέτει θόρυβο στα δείγματα και στη συνέχεια τα στέλνει στον λογαριασμό της edge impulse .

Μόλις τελειώσει η μεταφορά των δεδομένων ,ο σχεδιασμός του νευρωνικού γίνεται αποκλειστικά από το web interface του Edge Impulse .Στο σχήμα 3.9 παρατηρούμε τις επιλογές του edge impulse για την εξαγωγή χαρακτηριστικών και το σχεδιασμό νευρωνικού για την περίπτωση της αναγνώρισης λέξεων .



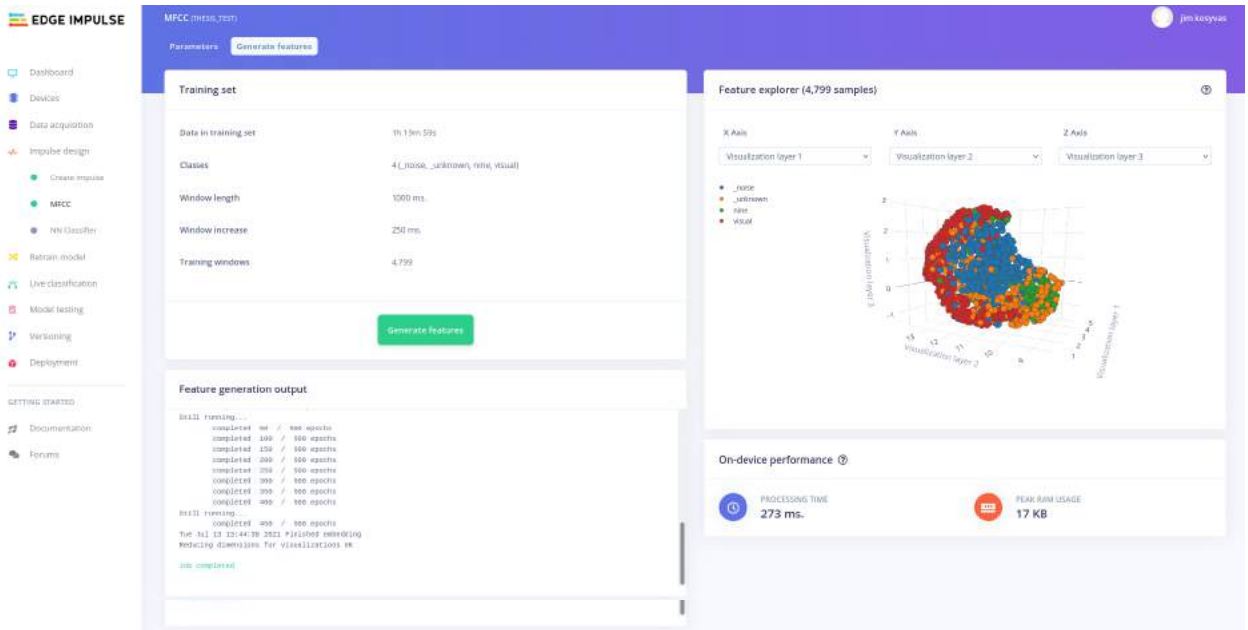
***Εικόνα 3.9**: Creating a neural network in edge impulse studio*

Στο κόκκινο μπλοκ ορίζεται η είσοδος του μοντέλου ,όπου για εμάς είναι είναι χρονοσειρές μεγέθους 1 sec και βήματος 250ms .Το επόμενο μπλοκ ορίζει το feature extraction και επιλέγουμε το mfcc καθώς ταιριάζει καλύτερα στη εφαρμογή μας .Τέλος στο κομμάτι της εκπαίδευσης διαλέγουμε ένα νευρωνικό που χρησιμοποιεί τη Keras βιβλιοθήκη .



***Εικόνα 3.10**: MFCC features*

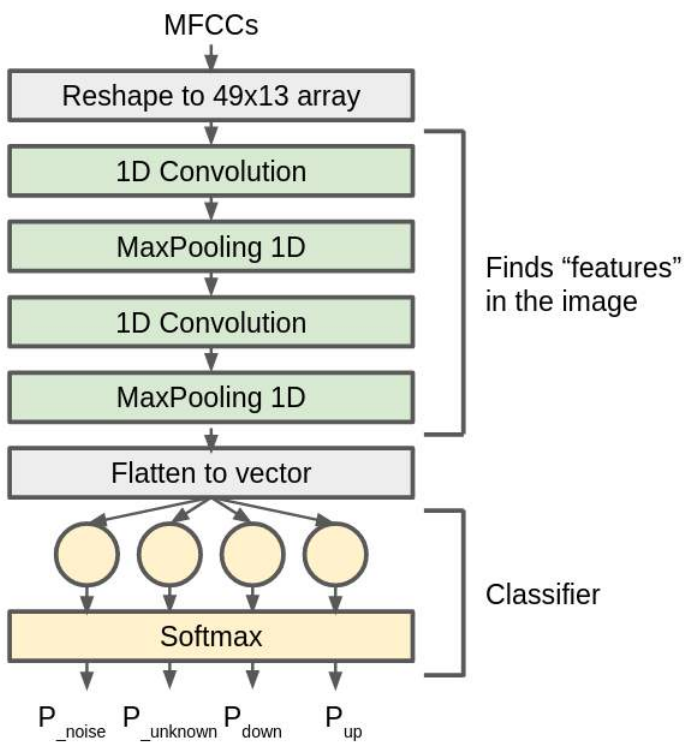
Στο επόμενο βήμα αναλύεται η μέθοδος MFCC στα δεδομένα ήχου και διακρίνονται τα spectrograms ,που παράγονται για κάθε διαφορετική λέξη .Επίσης δίνονται οι παράμετροι της μεθόδου mfcc και μπορούν να μεταβληθούν από τον χρήστη για να επιτύχει καλύτερο διαχωρισμό χαρακτηριστικών.Σε αυτό το σημείο γίνεται η παραγωγή των χαρακτηριστικών από το Edge Impulse όπως φαίνεται στο σχήμα 3.11 .



***Εικόνα 3.11**: MFCC extraction*

Ο κώδικας και η αρχιτεκτονική για το νευρωνικού που χρησιμοποιήθηκε για την εκπαίδευση φαίνεται παρακάτω :

```
# model architecture
model = Sequential()
model.add(Reshape((int(input_length / 13), 13), input_shape=(input_length, )))
model.add(Conv1D(8, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Conv1D(16, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(classes, activation='softmax', name='y_pred'))

# this controls the learning rate
opt = Adam(lr=0.005, beta_1=0.9, beta_2=0.999)
# this controls the batch size, or you can manipulate the tf.data.Dataset objects
yourself
BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder=False)
callbacks.append(BatchLoggerCallback(BATCH_SIZE, train_sample_count))

# train the neural network
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
model.fit(train_dataset, epochs=100, validation_data=validation_dataset, verbose=2,
callbacks=callbacks)
```

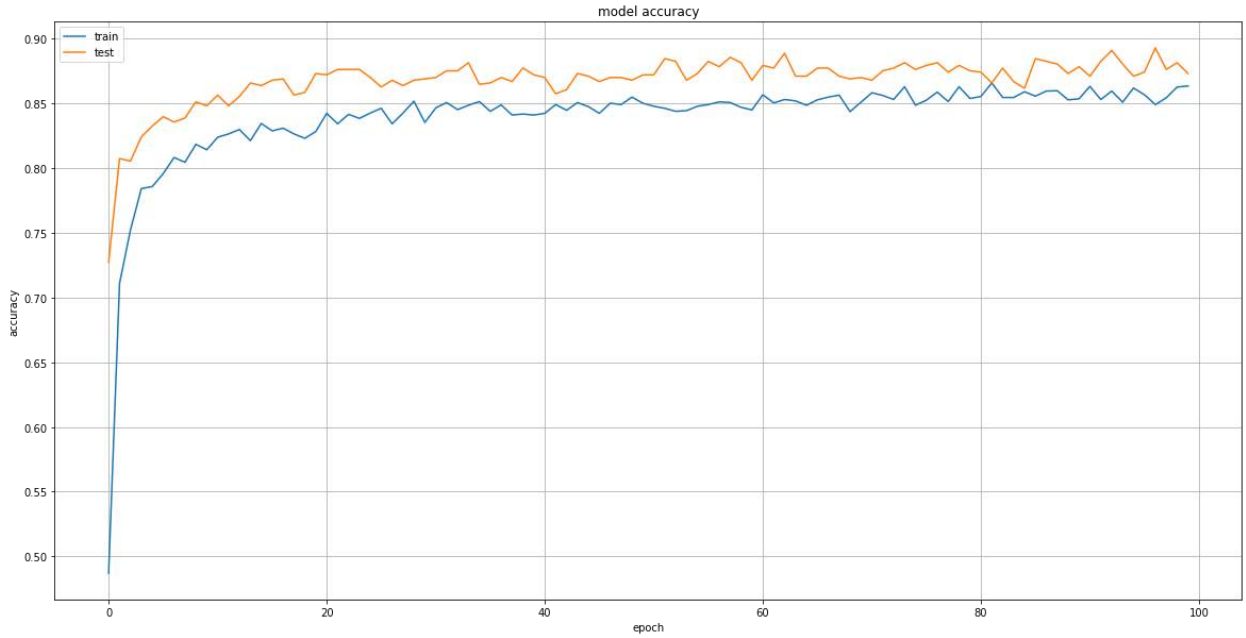


Εικόνα 3.12:*Model Architecture*

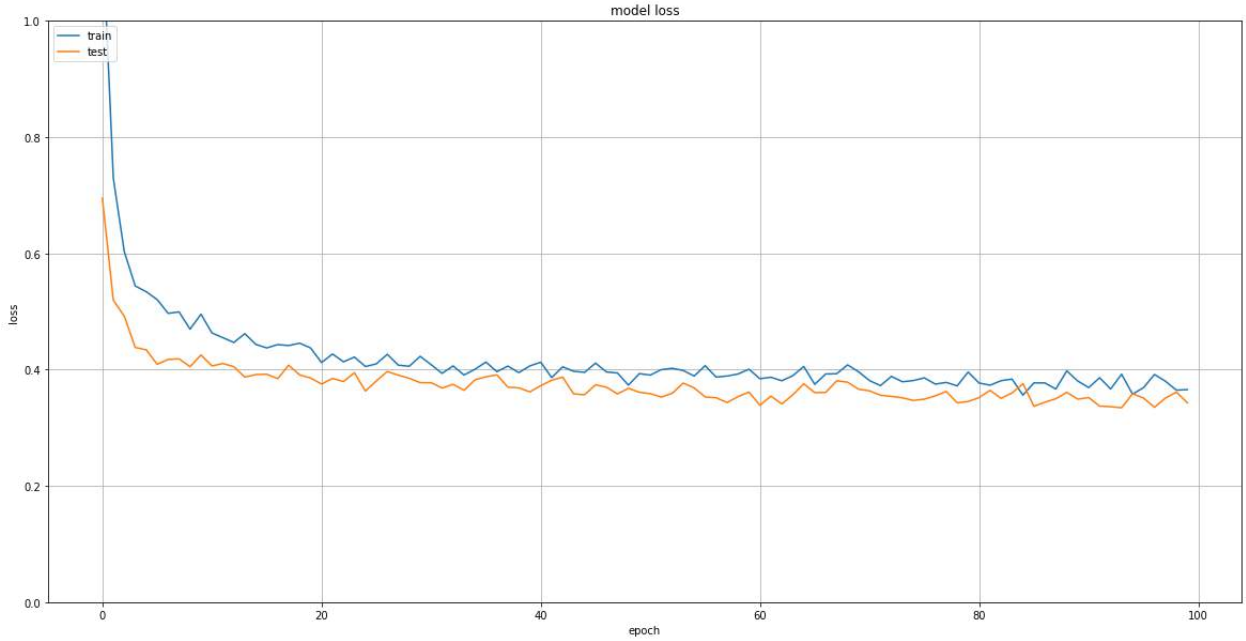
Μετά την εκπαίδευση του μοντέλου για τις ίδιες λέξεις του προηγούμενου παραδείγματος (nine,visual) ,το edge impulse παράγει τα παρακάτω αποτελέσματα :

***Εικόνα 3.13**:Edge impulse Results*

Παρατηρούμε ότι προκύπτουν εκτός από την ακρίβεια του μοντέλου και το confusion matrix κάποιες άλλες πληροφορίες σχετικά με την επίδοση στη συσκευή .Αυτά τα στοιχεία αντιστοιχούν στο arduino nano 33 ble sense με Cortex M4F 64Mhz ,το οποίο είναι η μόνη διαθέσιμη συσκευή προσομοιάζει καλύτερα το adafruit feather sense .



Εικόνα 3.14 *:Accuracy/ epochs edge impulse model*



Εικόνα 3.15 *:Loss/ epochs edge impulse model*

Τα παραπάνω διαγράμματα προκύπτουν από τα αποτελέσματα της εκπαίδευσης του νευρωνικού που περιγράφηκε και παρατηρείται ότι η συμπεριφορά του μοντέλου ως προς τα testing δεδομένα είναι σχετικά καλή .Αυτό σημαίνει ότι το μοντέλο είναι αξιόπιστο και για δείγματα ηχητικών λέξεων από διαφορετικούς ανθρώπους ,τα οποία δεν χρησιμοποιήθηκαν για την εκπαίδευση.

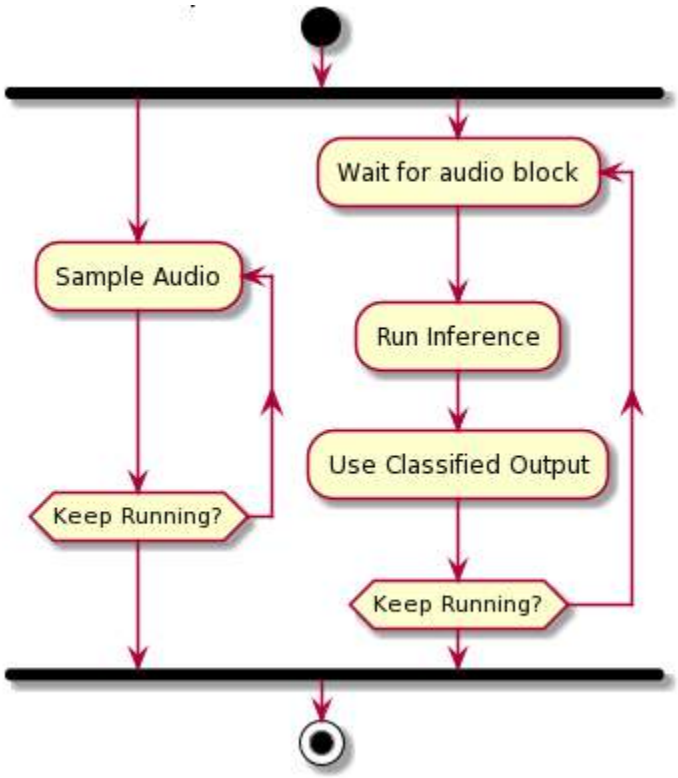
Το τελευταίο βήμα είναι το deployment όπου επιλέγουμε την εξαγωγή του προεκπαιδευμένου μοντέλου ως C++ βιβλιοθήκη .

3.2.1 Ενσωμάτωση edge impulse βιβλιοθήκης στο NRF-connect SDK

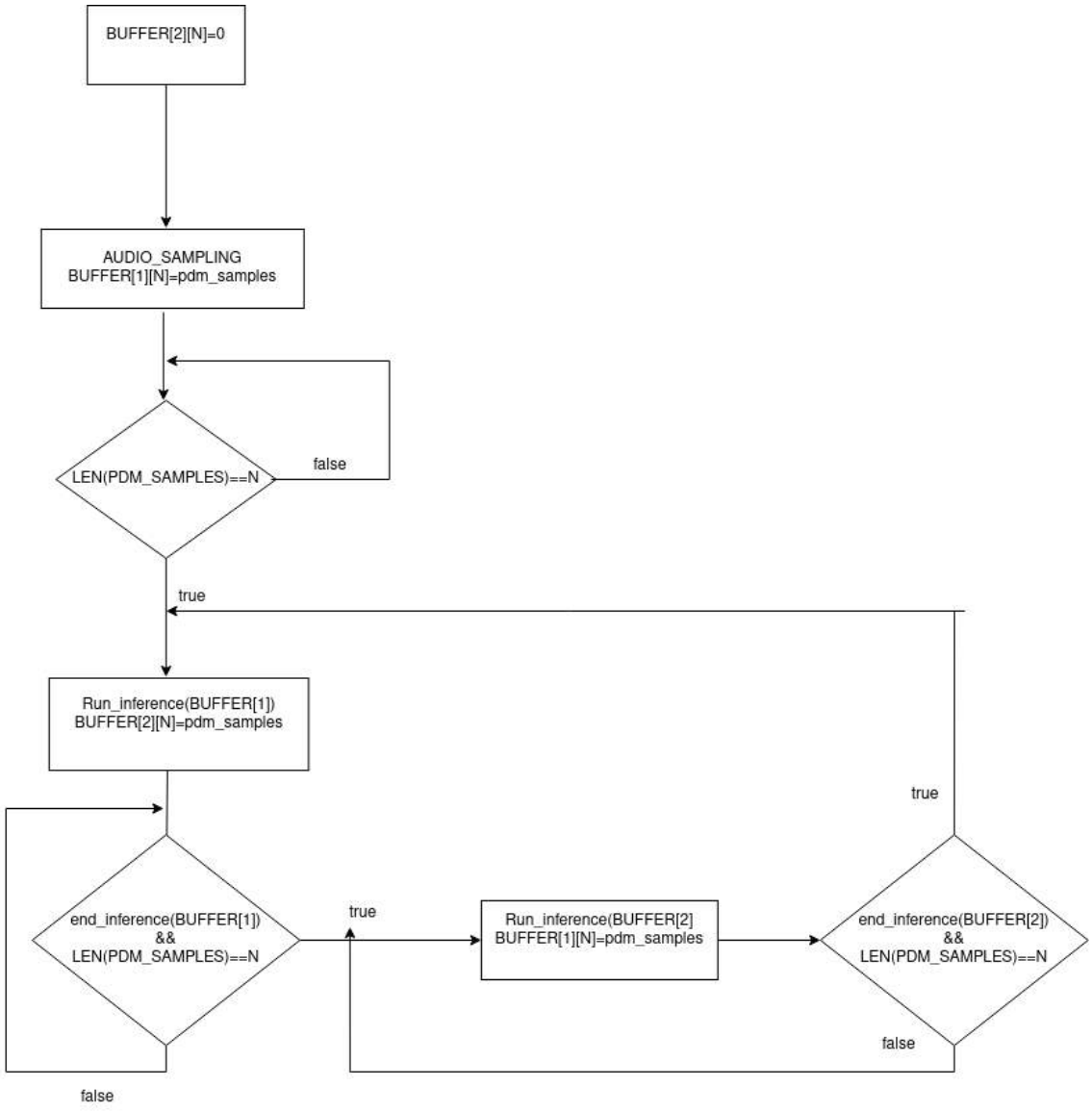
Η βιβλιοθήκη από το edge impulse περιλαμβάνει τα βάρη του προεκπαιδευμένου νευρωνικού δικτύου και το dsp block για την εξαγωγή των mfcc χαρακτηριστικών.Επομένως το μόνο που πρέπει να γίνει από πλευράς προγραμματιστή είναι η υλοποίηση της συνεχούς δειγματοληψίας του μικροφώνου και ο τρόπος που θα δίνεται σαν είσοδο στο μοντέλο .

Για αυτό το λόγο χρησιμοποιούμε τον μηχανισμό του double buffering όπου :

- 1 buffer χρησιμοποιείται για την δειγματοληψία του ήχου και παίρνει συνεχώς τις νέες τιμές του μικροφώνου
- 1 buffer για τη διαδικασία του inference ,όπου παίρνει τα δεδομένα ήχου κατόπιν εξάγονται οι συντελεστές mfcc και τρέχει το inference



Στην αρχή της δειγματοληψίας ο πρώτος buffer αποθηκεύει δείγματα ήχου και μόλις γεμίσει τότε χρησιμοποιείται για το inference και πλέον ο δεύτερος buffer δειγματοληπτεί ήχο .Σε κάθε επανάληψη αυτοί οι δύο buffers αλλάζουν ρόλους ώστε πάντα να υπάρχει ένας άδειος buffer για την αποθήκευση ήχου και ένας γεμάτος για το inference .Το μέγεθος των buffers αποτελεί καθοριστικό παράγοντα για την εφαρμογή καθώς σχετίζεται με το χρόνο καθυστέρησης και τη διαθέσιμη sram μνήμη του μικροελεγκτή .Για το λόγο αυτό το μέγεθος των buffers δεν αντιστοιχεί στον υπολογισμό ενός ολόκληρου δείγματος ήχου πχ 1 sec αλλά στα παράθυρα που χρειάζονται για το σχηματισμό ενός δείγματος (με βήμα 250ms έχουμε 4 παράθυρα). Έτσι προκύπτει ότι για 1 sec δείγματα οι buffers είναι πίνακες μεγέθους 4000.



Εικόνα 3.16 *:Διάγραμμα ροής για audio sampling*

Σχετικά με το προγραμματισμό του adafruit feather sense η διαδικασία είναι η ίδια που περιγράφηκε στο 3.1.5 με χρήση του Segger Jlink .

Παρακάτω φαίνεται η main.cpp που χρησιμοποιεί συναρτήσεις της βιβλιοθήκης της edge impulse για το inference .

```
89      while (true) {
90
91           gpio_pin_set(dev, PIN, (int)true);
92           gpio_pin_set(dev1, PIN1, (int)true);
93
94          bool m = ei_microphone_inference_record();
95          if (!m) {
96              ei_printf("ERR: Failed to record audio...\n");
97              break;
98          }
99          // the features are stored into flash, and we don't want to load everything into RAM
100         signal_t signal;
101         signal.total_length =EI_CLASSIFIER_SLICE_SIZE;
102         signal.get_data = &ei_microphone_audio_signal_get_data;
103         EI_IMPULSE_ERROR res = run_classifier_continuous(&signal, &result,false);
104
105         // Print output predictions (once every 4 predictions)
106     if(++print_results >= (EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW >> 1))
107     {
108       // Comment this section out if you don't want to see the raw scores
109       ei_printf("\n Predictions (DSP: %d ms, NN: %d ms)\r\n", result.timing.dsp, result.timing.classification);
110       for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++)
111       {
112
113         ei_printf("    %s: %.5f\r\n", result.classification[ix].label, result.classification[ix].value);
114
115
116       }
117       print_results = 0;
118     }
119
120
121         if(result.classification[2].value>0.6){
122             printk("nine found \n");
123             gpio_pin_set(dev, PIN, (int)false);
124             k_msleep(200);
125
126
127             }
128
129         if(result.classification[3].value>0.6 ){
130              gpio_pin_set(dev1, PIN1, (int)false);
131              printk("visual found \n");
132              k_msleep(200);
133
134          }
135
136     }
137
138     ei_microphone_inference_end();
139
140 }
```

Στο κώδικα της main ,δειγματοληπτούμε τον ήχο για παράθυρο 250ms κάθε φορά (ei_microphone_inference_record()) και τον δίνουμε ως είσοδο στην run_classifier_continuous() , η οποία υπολογίζει τους mfcc συντελεστές για όλο το δείγμα του 1 sec και κάνει inference .Το αποτέλεσμα είναι η πιθανότητα εύρεσης της κάθε λέξης μαζί με τους χρόνους που απαιτήθηκαν για την ψηφιακή επεξεργασία των δειγμάτων και το inference .

```
Predictions (DSP: 125 ms, NN: 6 ms)
   _noise: 0.26758
   _unknown: 0.23047
   nine: 0.25977
   visual: 0.24219

Predictions (DSP: 124 ms, NN: 6 ms)
   _noise: 0.11133
   _unknown: 0.26562
   nine: 0.16406
   visual: 0.45898

Predictions (DSP: 124 ms, NN: 7 ms)
   _noise: 0.00000
   _unknown: 0.48438
   nine: 0.02734
   visual: 0.48828

Predictions (DSP: 124 ms, NN: 6 ms)
   _noise: 0.76172
   _unknown: 0.22461
   nine: 0.00195
   visual: 0.00781

Predictions (DSP: 124 ms, NN: 6 ms)
   _noise: 0.97266
   _unknown: 0.02148
   nine: 0.00391
   visual: 0.00195
```

Εικόνα 3.17 *:Uart Output from nrf52840*

Στο παράδειγμα θεωρούμε ότι αν η πιθανότητα είναι μεγαλύτερη του 0.6 η λέξη έχει βρεθεί και αντίστοιχα αναβοσβήνουν τα led για οπτική επιβεβαίωση .

Παρακάτω στο πίνακα φαίνονται τα χαρακτηριστικά του προεκπαιδευμένου μοντέλου από το edge impulse αφού χτιστεί από το nrf-connect.

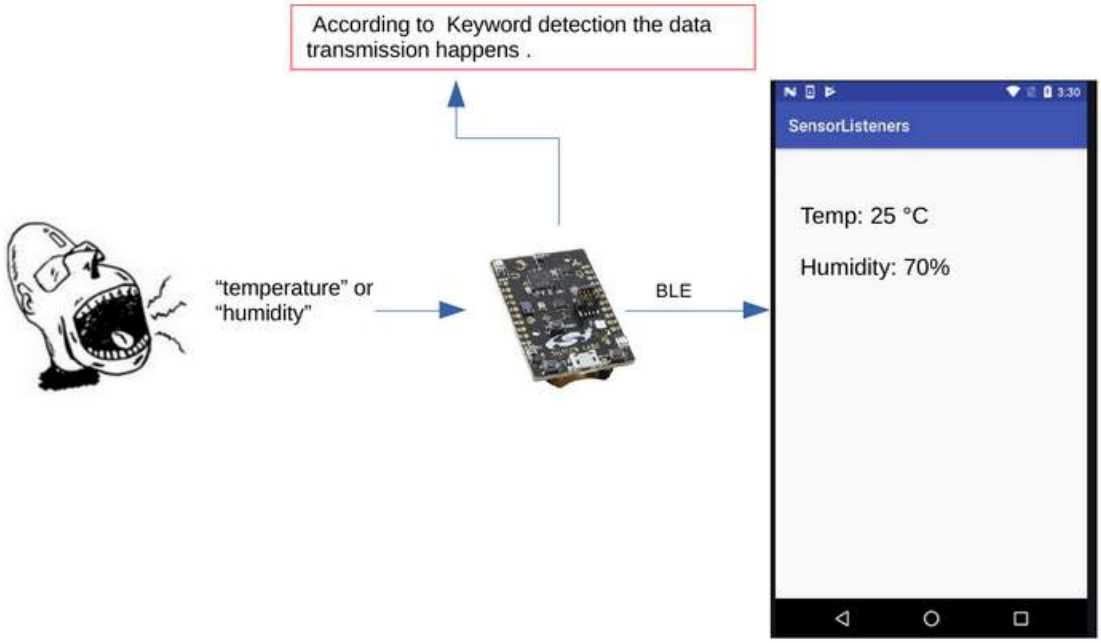
words	quantized model accuracy	flash	sram
nine , visual	88.2 %	289 kB	39 kB

3.3 Συμπεράσματα

Συγκρίνοντας τις δύο παραπάνω μεθόδους διαπιστώθηκε ότι παρά την μεγαλύτερη ακρίβεια που εμφανίζει το νευρωνικό της πρώτης στη πραγματικότητα η επίδοση του μοντέλου στη συσκευή είναι χειρότερη .Αυτό συμβαίνει γιατί παρατηρούνται πολλά false positives περιπτώσεις όπου φαίνεται ότι ο μοντέλο βρίσκει τη λέξη δίχως να έχει ειπωθεί κάτι από το χρήστη .Ένας πιθανός λόγος για την εμφάνιση αυτού του σφάλματος είναι η χρήση διαφορετικού pipeline για την δειγματοληψία του ήχου στα δύο μοντέλα.Για το λόγο επιλέχθηκε η δεύτερη μέθοδος με την βιβλιοθήκη της edge impulse για την περαιτέρω ανάπτυξη της εφαρμογής ,που αναλύεται στο επόμενο κεφάλαιο .

4. Προσθήκη Περιφερειακών Λειτουργιών

Για να ολοκληρωθεί η αρχική ιδέα της εφαρμογής πρέπει να προστεθούν οι ενέργειες που θα πραγματοποιούνται στη περίπτωσης εύρεσης κάποια λέξης .Αυτές οι ενέργειες θα είναι η δειγματοληψία ενός αισθητήρα θερμοκρασίας ,υγρασίας και η αποστολή των τιμών μέσω Bluetooth σε μια εφαρμογή ενός smartphone (παρακάτω σχήμα).



Εικόνα 4.1 *: General information pipeline*

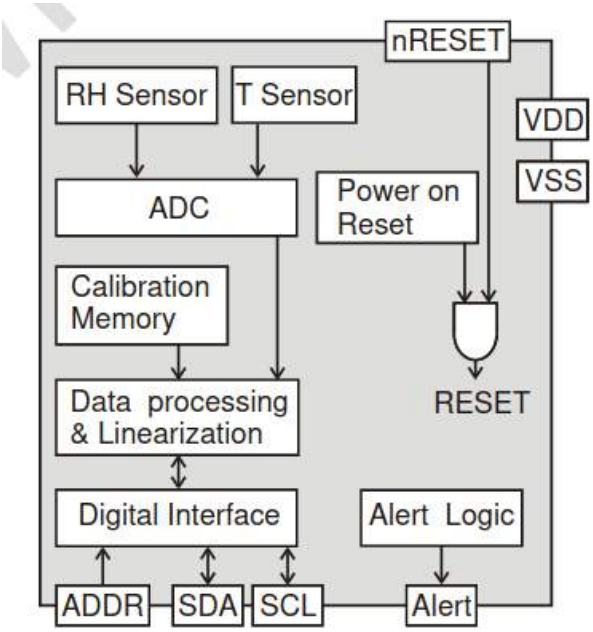
4.1 SHT3XD Αισθητήρας

Ο SHT3XD είναι αισθητήρας θερμοκρασίας και υγρασίας ,ο οποίος είναι ενσωματωμένος στο adafruit feather sense και έχει τα εξής χαρακτηριστικά :

- πλήρη βαθμονομημένος με ψηφιακή μορφή εξόδου
- τροφοδοσία λειτουργίας από 2.4 έως 5.5 volt
- χρησιμοποιεί I2C πρωτόκολλο επικοινωνίας με ταχύτητα μετάδοσης μέχρι 1Mhz
- σχετική ακρίβεια ± 2%RH για υγρασία και ±0.3 Celcius για θερμοκρασία
- μεγάλη ταχύτητα ενεργοποίησης και μετρήσεων

Παρακάτω φαίνεται ο αισθητήρας πάνω στο μικροελεγκτή μαζί με το λειτουργικό του διάγραμμα .

Εικόνα 4.2 *:Position of sensor in device with red color*



Εικόνα 4.3 *:Functional sensor block diagram*

4.1.1 Δειγματοληψία Μετρήσεων

Στο παρακάτω παράδειγμα δειγματοληπτούμε τον αισθητήρα SHT3XD με συχνότητα 2Hz χρησιμοποιώντας τον ήδη υπάρχων driver του zephyr RTOS και εκτυπώνεται η αντίστοιχη θερμοκρασία και υγρασία του περιβάλλοντος .

```
#include <zephyr.h>
#include <device.h>
#include <drivers/sensor.h>
#include <stdio.h>
```

```
#include <drivers/uart.h>

void main(void)
{
   const struct device *dev = device_get_binding("SHT3XD");
   int rc;

   if (dev == NULL) {
            printf("Could not get SHT3XD device\n");
            return;
   }

           while (true) {
            struct sensor_value temp, hum;

            rc = sensor_sample_fetch(dev);
            if (rc == 0) {
                     rc = sensor_channel_get(dev, SENSOR_CHAN_AMBIENT_TEMP,
                                                &temp);
            }
            if (rc == 0) {
                     rc = sensor_channel_get(dev, SENSOR_CHAN_HUMIDITY,
                                                &hum);
            }
            if (rc != 0) {
                     printf("SHT3XD: failed: %d\n", rc);
                     break;
            }

            printf("SHT3XD: %.2f Cel ; %0.2f %%RH\n",
                    sensor_value_to_double(&temp),
                    sensor_value_to_double(&hum));

            k_sleep(K_MSEC(2000));
   }
}
```

```
SHT3XD: 26.27 Cel ; 41.73 %RH
SHT3XD: 26.27 Cel ; 41.74 %RH
SHT3XD: 26.27 Cel ; 41.69 %RH
SHT3XD: 26.26 Cel ; 41.97 %RH
SHT3XD: 26.24 Cel ; 41.91 %RH
SHT3XD: 26.26 Cel ; 41.89 %RH
SHT3XD: 26.26 Cel ; 41.82 %RH
SHT3XD: 26.26 Cel ; 41.78 %RH
SHT3XD: 26.24 Cel ; 41.76 %RH
SHT3XD: 26.24 Cel ; 41.77 %RH
SHT3XD: 26.23 Cel ; 41.76 %RH
SHT3XD: 26.24 Cel ; 41.81 %RH
SHT3XD: 26.21 Cel ; 41.85 %RH
SHT3XD: 26.21 Cel ; 41.88 %RH
```

Εικόνα 4.4 *:Temp/humidity values from sensor*

4.2 Bluetooth Low Energy

Το Bluetooth είναι ένα πρότυπο ασύρματης τεχνολογίας. Η τεχνολογία Bluetooth αναπτύχθηκε ως τρόπος ανταλλαγής δεδομένων σε μικρό χρονικό διάστημα, κοντινή απόσταση και χωρίς την ανάγκη για καλώδια.

Το Bluetooth Low Energy (BLE), που μερικές φορές αναφέρεται ως Bluetooth Smart, είναι ένα ελαφρύ υποσύνολο κλασικού Bluetooth και εισήχθη ως μέρος της προδιαγραφής Bluetooth 4.0. Παρόλο που υπάρχει κάποια σχέση με το κλασικό Bluetooth, το BLE έχει στην πραγματικότητα μια εντελώς διαφορετική γενεά και ξεκίνησε από τη Nokia ως ένα εσωτερικό έργο που ονομάζεται Wibree, προτού εγκριθεί από το Bluetooth SIG (Townsend,2018).

Εξετάζοντας τη διαφορά μεταξύ Bluetooth και Bluetooth Low Energy (BLE), είναι σημαντικό να γίνει αναφορά στην κατανάλωση ενέργειας. Το Bluetooth σχεδιάστηκε αρχικά για συνεχείς εφαρμογές ροής δεδομένων. Αυτό σημαίνει ότι μπορεί να γίνει ανταλλαγή πολλών δεδομένων σε κοντινή απόσταση. Το BLE χρησιμοποιείται για λύσεις με πολύ χαμηλή κατανάλωση ενέργειας και του «Διαδικτύου των Πραγμάτων» (“Internet of Things” - IoT) με στόχο συσκευές χαμηλού κόστους που λειτουργούν με μπαταρίες και μπορούν να συνδεθούν γρήγορα και να σχηματίσουν απλούς ασύρματους συνδέσμους .

Το BLE λειτουργεί στη ζώνη συχνοτήτων ISM 2,4 GHz, όπως και το Bluetooth. Όμως, σε αντίθεση με την κλασσική τεχνολογία Bluetooth, το BLE παραμένει σε κατάσταση αναστολής συνεχώς, εκτός από την περίπτωση που ξεκινά μια σύνδεση. Οι πραγματικοί χρόνοι σύνδεσης είναι μόνο μερικά ms, σε αντίθεση με το Bluetooth που θα πάρει ~ 100ms.

Ο λόγος για τον οποίο οι συνδέσεις είναι τόσο σύντομες είναι ότι οι ρυθμοί δεδομένων είναι πολύ υψηλοί στα 1 Mb/s .

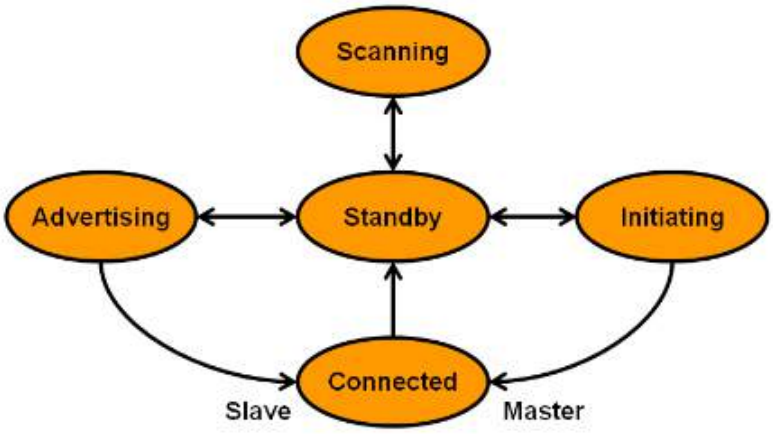
Συνοπτικά, το Bluetooth και το BLE χρησιμοποιούνται για πολύ διαφορετικούς σκοπούς.Το Bluetooth μπορεί να χειριστεί πολλά δεδομένα, αλλά καταναλώνει γρήγορα τη ζωή της μπαταρίας. Το BLE χρησιμοποιείται για εφαρμογές που δεν χρειάζεται να ανταλλάσσουν μεγάλες ποσότητες δεδομένων και συνεπώς μπορούν να λειτουργούν με μπαταρία για χρόνια με φθηνότερο κόστος.

4.2.1 BLE Αρχιτεκτονική

Οι συσκευές BLE μπορούν να βρίσκονται σε διαφορετικές καταστάσεις λειτουργίας(operating states) και ρόλους (roles) ανάλογα με τη λειτουργία τους. Επομένως, οι πιθανές καταστάσεις είναι οι εξής:

- Αναμονή (standby): Δεν μεταδίδει ή λαμβάνει πακέτα.
- Διαφήμιση (advertising): Μεταδίδει διαφημίσεις σε κανάλια διαφήμισης.
- Σάρωση (scanning): Αναζητά διαφημιστές.
- Έναρξη (initiating): Ξεκινά σύνδεση με ένα διαφημιζόμενο.
- Σύνδεση (Connection):
    - Κύριος ρόλος (master role): Επικοινωνεί με τη συσκευή στο ρόλο σκλάβου.
    - Ρόλος σκλάβου (slave role): Επικοινωνεί με μια συσκευή στον κύριο ρόλο.

Εικόνα 4.5 *: BLE LINK LAYER STATE MACHINE*



Η τοπολογία δικτύου του BLE είναι ο τύπος αστεριού. Οι κύριες συσκευές μπορούν να έχουν πολλαπλές συνδέσεις στρώματος συνδέσμου με περιφερειακά (σκλάβους) και ταυτόχρονα να ανιχνεύουν άλλες συσκευές. Από την άλλη πλευρά, ένας σκλάβος μπορεί να έχει μόνο μια σύνδεση στρώματος συνδέσμου σε έναν κύριο. Η στοίβα πρωτοκόλλου BLE χωρίζεται σε τρία βασικά μέρη: ελεγκτής (controller), κεντρικός υπολογιστής (host) και εφαρμογές (applications). Ο ελεγκτής είναι μια φυσική συσκευή που μπορεί να μεταδίδει και να λαμβάνει ραδιοσήματα και τα ερμηνεύει ως πακέτα με πληροφορίες. Περιλαμβάνει το Φυσικό Επίπεδο (Physical Layer), την Απευθείας Δοκιμή (Direct Test Mode), το Επίπεδο Σύνδεσης (Layer Link) και τον κεντρικό υπολογιστή για τη ως διασύνδεση ελεγκτή ("Host Controller Interface" - HCI). Ο κεντρικός υπολογιστής (host) είναι μια στοίβα λογισμικού που διαχειρίζεται τον τρόπο επικοινωνίας μεταξύ δύο ή περισσοτέρων συσκευών. Δεν υπάρχει καθορισμένη ανώτερη διεπαφή για τον κεντρικό υπολογιστή, κάθε λειτουργικό σύστημα ή περιβάλλον έχει διαφορετικό τρόπο έκθεσης API φιλοξενίας για προγραμματιστές.

Περιέχει το πρωτόκολλο ελέγχου λογικής σύνδεσης και προσαρμογής (“Logical Link Control and Adaptation Protocol” – L2CAP), τον διαχειριστή ασφαλείας (Security Manager), το πρωτόκολλο χαρακτηριστικών (“Attribute protocol” - ATT), το γενικό προφίλ χαρακτηριστικών (“Generic Attribute Profile” - GATT) και το προφίλ γενικής πρόσβασης (“Generic Access Profile” - GAP).Οι Εφαρμογές χρησιμοποιούν τη στοίβα λογισμικού και επομένως τον ελεγκτή, για να ενεργοποιήσετε μια περίπτωση χρήσης. Το Επίπεδο Εφαρμογής ορίζει τρεις τύπους προδιαγραφών: χαρακτηριστικό, υπηρεσία και προφίλ .

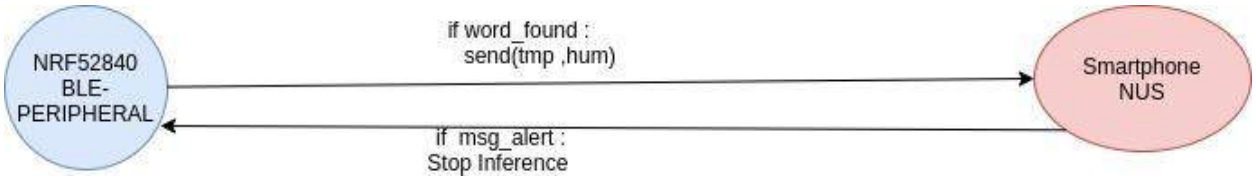
Εφαρμογές (Applications)	
Apps	
GAP	
GATT	
ATT	Security Manager (διαχειριστή ασφαλείας)
Host	
HCI	
Link Layer (επίπεδο σύνδεσης)	
Physical Layer (φυσικό επίπεδο)	Direct Test Mode (λειτουργία απευθείας δοκιμής)

Στοίβα Πρωτοκόλλου BLE

4.2.2 Προσθήκη BLE Λειτουργίας σε Adafruit Feather Sense

Για την επικοινωνία του μικροελεγκτή με smartphone μέσω BLE , προγραμματίζουμε το adafruit feather σε ρόλο peripheral δηλαδή κάνει advertising μηνύματα και συνδέεται ως σκλάβος (slave).

Από την πλευρά του smartphone χρησιμοποιήθηκε η εφαρμογή της nordic nRF Toolbox App ,η οποία με την υπηρεσία Nordic Uart Service (NUS) μπορεί να στέλνει και να δέχεται μηνύματα μέσω UART .Η NUS ορίζει δύο χαρακτηριστικά ,το RX (“write” ιδιότητες) και το TX (“notify” ιδιότητες) κανάλι για τη ρύθμιση UART επικοινωνίας .



Εικόνα 4.6 *: BLE communication flow*

Παρακάτω φαίνεται η τελική μορφή της συνάρτησης με την προσθήκη της μετάδοσης δεδομένων μέσω BLE από το μικροελεγκτή στο smartphone .

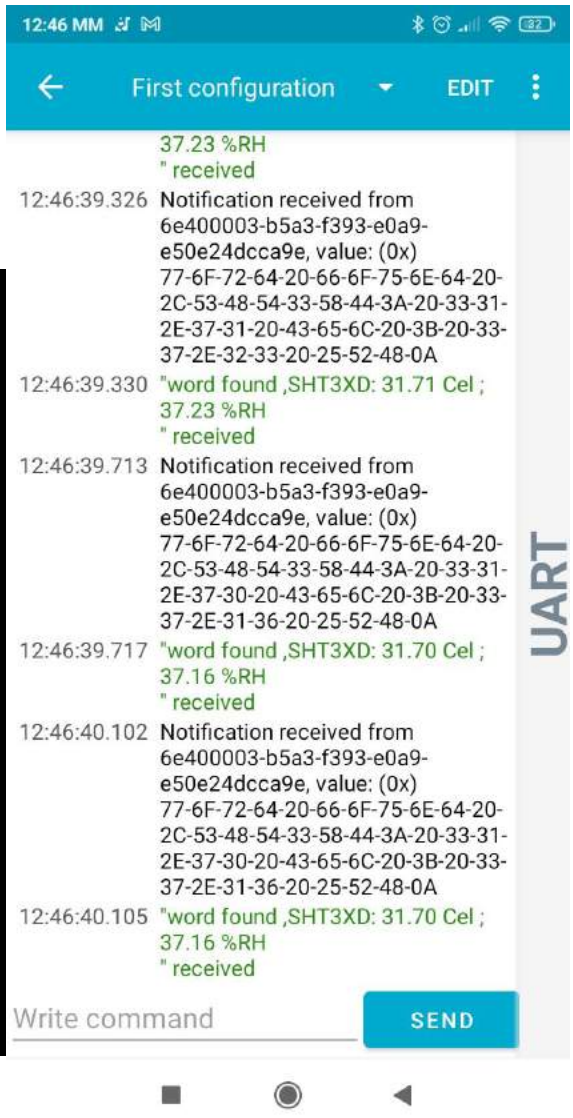
```
void main (void) {
        char ble_printf[64] = {0};
        printk("app starting \n");
        ble_nus_init();
        int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);
        run_classifier_init();
        ei_microphone_inference_start(EI_CLASSIFIER_SLICE_SIZE); //allocate memory for inference buffers
        pdm_init();                                 // initialize the PDM microphone for the Adafruit sense board
        nrfx_pdm_start();                                   //start sampling audio from the microphone
        ei_impulse_result_t result = { 0 };
        double b1=0,b2=0;


        while (true) {

        bool m = ei_microphone_inference_record();
        if (!m) {
        printk("ERR: Failed to record audio...\n");
        break;
        }
        // the features are stored into flash, and we don't want to load everything into RAM
        signal_t signal;
        signal.total_length =EI_CLASSIFIER_SLICE_SIZE;
        signal.get_data = &ei_microphone_audio_signal_get_data;
```

```
        EI_IMPULSE_ERROR res = run_classifier_continuous(&signal, &result,false);

        // Print output predictions (once every 4 predictions)
        if(++print_results >= (EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW >> 1))
        {
        printk("\n Predictions (DSP: %d ms, NN: %d ms)\r\n", result.timing.dsp, result.timing.classification);
        for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++)
        {

        printk(" %s: %.5f\r\n", result.classification[ix].label, result.classification[ix].value);


        }
        print_results = 0;
        }


        if(result.classification[2].value>0.6){
        tmp(b1,b2);
        sprintf(ble_printf,"SHT3XD: %.2f Cel ; %0.2f %%RH\n",b1,b2);
        ble_nus_send_data(ble_printf, strlen(ble_printf));
        k_msleep(200);


        }

        if(result.classification[3].value>0.6 ){
        tmp(b1,b2);
        sprintf(ble_printf,"SHT3XD: %.2f Cel ; %0.2f %%RH\n",b1,b2);
        ble_nus_send_data(ble_printf, strlen(ble_printf));
        k_msleep(200);

        }
        if(ei_ble_user_invoke_stop()) {
        printk("Inferencing stopped by user\r\n");
        break;
        }
        }

}
```

Model: BLE & KWS	quantized model accuracy	flash	sram
Words :nine , visual	88.2 %	471 kB	71 kB

Παρακάτω παρουσιάζονται τα αποτελέσματα της εφαρμογής μέσω UART από το μικροελεγκτή και από την εφαρμογή nrf-toolbox στο smartphone .

```
Predictions (DSP: 122 ms., Classification: 5 ms., Anomaly: 0 ms.):
    _noise: 0.47656
    _unknown: 0.16211
    nine: 0.02734
    visual: 0.33203
Predictions (DSP: 122 ms., Classification: 5 ms., Anomaly: 0 ms.):
    _noise: 0.22852
    _unknown: 0.05469
    nine: 0.00195
    visual: 0.71289
Predictions (DSP: 122 ms., Classification: 5 ms., Anomaly: 0 ms.):
    _noise: 0.89062
    _unknown: 0.05078
    nine: 0.05078
    visual: 0.00977
Predictions (DSP: 122 ms., Classification: 4 ms., Anomaly: 0 ms.):
    _noise: 0.65625
    _unknown: 0.09375
    nine: 0.23633
    visual: 0.01367
Predictions (DSP: 122 ms., Classification: 4 ms., Anomaly: 0 ms.):
    _noise: 0.95703
    _unknown: 0.03320
    nine: 0.00586
    visual: 0.00391
Predictions (DSP: 123 ms., Classification: 4 ms., Anomaly: 0 ms.):
    _noise: 0.96875
    _unknown: 0.01562
    nine: 0.00586
    visual: 0.00977
```



Εικόνα 4.7 *:Uart Output of nrf52840 & ble output from*

Όταν ειπωθεί οποιαδήποτε από τις δύο λέξεις (nine,visual) για τις οποίες εκπαιδεύτηκε το μοντέλο ,τότε στέλνονται οι τιμές θερμοκρασίας και υγρασίας μέσω BLE στην εφαρμογή του κινητού.Επίσης ο χρήστης μέσα από την εφαρμογή μπορεί να σταματήσει την λειτουργία της εφαρμογής στέλνοντας το μήνυμα “stop” .

5. Αποτελέσματα Εφαρμογής

5.1 Σύγκριση Νευρωνικών Μοντέλων

Σε αυτό το σημείο χρησιμοποιώντας το tuner εργαλείο της edge impulse θα δημιουργηθούν διαφορετικές αρχιτεκτονικές νευρωνικών μεταβάλλοντας τις hyperparameters του αρχικού μοντέλου και θα μελετηθει η συμπεριφορά τους ως προς τη κατανάλωση μνήμης ,την επίδοση της εφαρμογής και τους χρόνους επεξεργασίας .

Στη συνέχεια θα εξεταστεί πόσο επηρεάζεται η ακρίβεια και η επίδοση αυτών των μοντέλων αυξάνοντας τις λέξεις προς αναγνώριση.

Οι παράμετροι του CNN που θα μεταβάλλονται για την εύρεση του καλύτερου μοντέλου για την εφαρμογή μας είναι οι παρακάτω :

- Ο αριθμός των φίλτρων στα συνελικτικά στρώματα
- Το μέγεθος των φίλτρων
- Το μέγεθος του dense στρώματος
- Dropout rate
- Ο αριθμός των συνελικτικών στρωμάτων

Ενώ το learning rate διατηρήθηκε σταθερό στα 0.05 και ο αριθμός των επαναλήψεων στις 100 .

Model ID	FilterNum1	FilterNum2	FilterNum3	DenseSize	Dropout Rate	FilterSize	Accuracy [%]	Processing Time on Device (ms)	Number of Words
mfcc-conv 1d-56b	16	32	-	-	0.6	2x3	91.7	80	2
mfcc-conv 1d-941	16	32	64	64	0.25	3x3	90.6	90	2
mfcc-conv 1d-6ee	8	16	-	64	0.25	2x3	87.2	62	2
mfcc-conv 2d-363	8	16	32	-	0.25	3x3	86.4	92	2
mfcc-conv 1d-5d0	16	32	64	-	0.5	3x3	84.7	65	3
mfcc-conv 1d-679	16	32	-	-	0.5	2x3	85.3	80	3
mfcc-conv 1d-85d	8	16	-	64	0.25	2x3	82.8	63	3
mfcc-conv 1d-93f	16	32	64	-	0.5	3x3	73.9	65	10

Όπως φαίνεται από το παραπάνω πίνακα όσο αυξάνεται ο αριθμός των λέξεων τόσο μειώνεται η ακρίβεια του νευρωνικού .Επίσης σημαντική παράμετρο για την επιλογή του

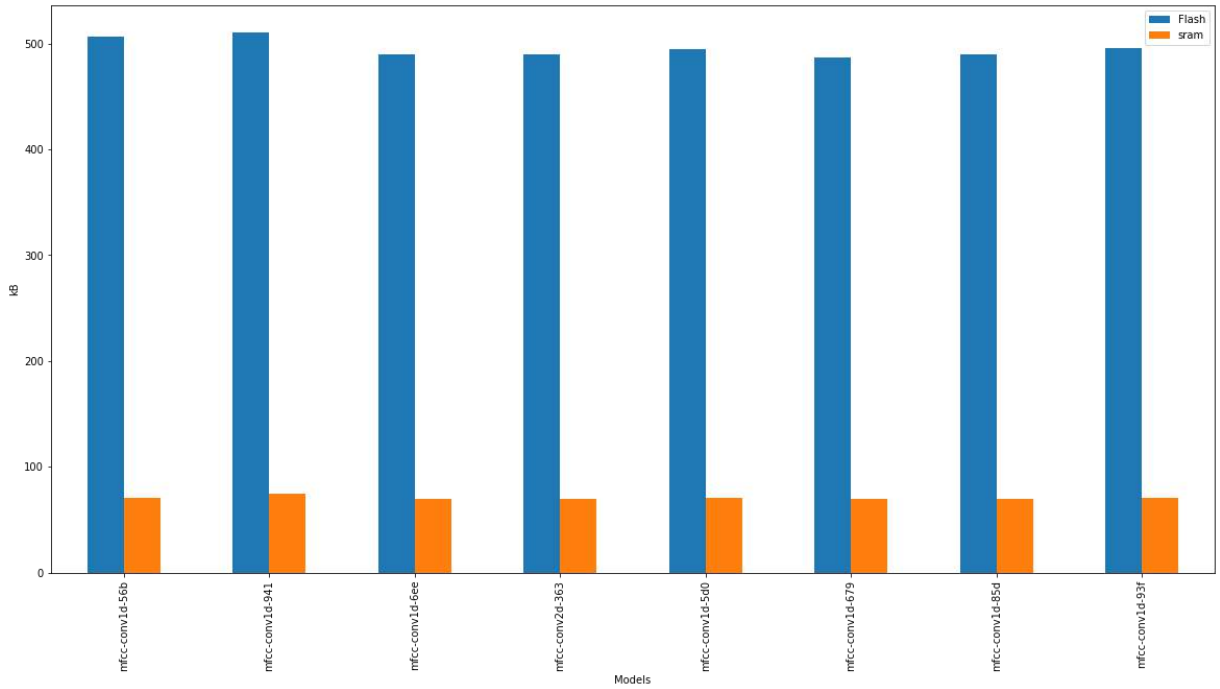
κατάλληλου μοντέλου είναι η ελαχιστοποίηση της καθυστέρησης (latency) καθώς επηρεάζει σε μεγάλο βαθμό την επίδοση της εφαρμογής στον μικροελεγκτή .Για αυτό το ιδανικό μοντέλο πρέπει να συνδυάζει μεγάλη ακρίβεια με καλή επίδοση on device δηλαδή μικρή καθυστέρηση υπολογισμών .

5.1.1 Memory Footprint

Για να υπολογίσουμε τη κατανάλωση μνήμης των μοντέλων θα αναλύσουμε το μέγεθος της Flash και Ram στα δυαδικά αρχεία ,που μεταγλωτιστηκαν για testing στον μικροελεγκτή .Στο zephyr RTOS μόλις ολοκληρωθεί το χτίσιμο της εφαρμογής παράγονται αυτόματα το ποσοστό χρήσης Flash και Ram μνήμης όπως φαίνεται στο παρακάτω παράδειγμα .

```
$ west build -b adafruit_feather_nrf52840_sense
……...................................................................
Memory region         Used Size  Region Size      Used
        FLASH:        473500 B      1 MB          45.16%
        SRAM:         71360 B       256 KB        27.22%
```

Όλα τα μεγέθη μνήμης για τα μοντέλα που χρησιμοποιήθηκαν στο 5.1 φαίνονται στο παρακάτω ραβδόγραμμα .



Εικόνα 5.1 *:Bar graph for Memory Consumption*

Από το διάγραμμα παρατηρούμε ότι δεν υπάρχουν μεγάλες διαφορές στη χρησιμοποιούμενη μνήμη στα παραπάνω μοντέλα .Έτσι για τον μικροελεγκτή nrf52840 για τις ανάγκες της εφαρμογής αναγνώρισης λέξεων μαζί με τη προσθήκη BLE χρησιμοποιείται περίπου το 50% της flash και το 28% της sram του.Αυτό σημαίνει ότι υπάρχει η δυνατότητα προσθήκης και άλλων μοντέλων όπως αναγνώρισης κίνησης (fall detection) για να αξιοποιηθεί η διαθέσιμη μνήμη .

5.1.2 Benchmarks

Σε αυτό το σημείο τα παραπάνω μοντέλα γίνονται deploy σε

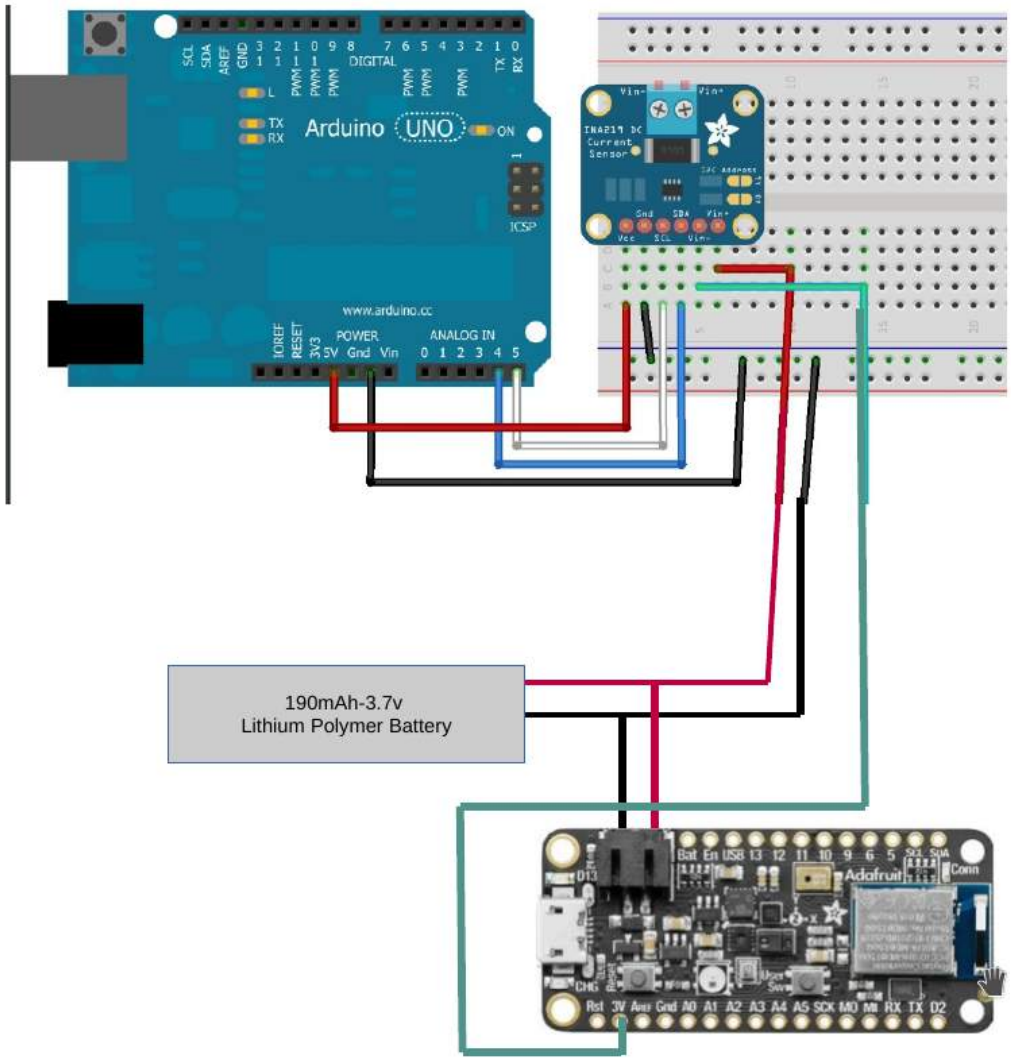
- Raspberry pie 4
- Jetson Tx2
- Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
- Intel(R)Core(TM) i3-7100CPU@3.90GHz

Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz			**RaspberryPi4 (aarch64)**		**JetsonTx2 (aarch64)**		**Intel(R)Core(TM) i3-7100CPU@3.90GHz**	
Model ID	Time(ms)	Exec Size(mB)	Time(ms)	Exec Size(mB)	Time(ms)	Exec Size(mB)	Time(ms)	Exec Size(mB)
mfcc-conv1d-56b	2	7.48	6	8.18	5	9.34	3	7.48
mfcc-conv1d-941	2	7.53	6	8.22	5	9.46	3	7.53
mfcc-conv1d-6ee	2	7.45	6	8.19	5	9.30	3	7.45
mfcc-conv2d-363	2	7.49	5	8.21	5	9.28	4	7.49
mfcc-conv1d-5d0	2	7.50	6	8.20	5	9.27	3	7.50
mfcc-conv1d-679	2	7.48	5	8.18	4	9.22	3	7.48
mfcc-conv1d-85d	2	7.49	5	8.19	4	9.21	3	7.49
mfcc-conv1d-93f	3	7.51	5	8.20	4	9.23	4	7.51

Παρατηρούμε ότι παρόλο που ο χρόνος επεξεργασίας μειώνεται σημαντικά καθώς τρέχουμε την εφαρμογή σε συσκευές μεγαλύτερης υπολογιστικής ισχύς ,το μέγεθος της αυξάνεται αρκετά .Για την δειγματοληψία του ήχου στις linux συσκευές χρησιμοποιήθηκε η ALSA (Advanced Linux Sound Architecture) βιβλιοθήκη .

5.1.3 Power profiling of system

Για τον υπολογισμό της καταναλούμενης ισχύος της εφαρμογής που τρέχει στον μικροελεγκτή nrf52840 χρησιμοποιήθηκε η παρακάτω διάταξη .



Εικόνα 5.2 *:Measuring Setup*

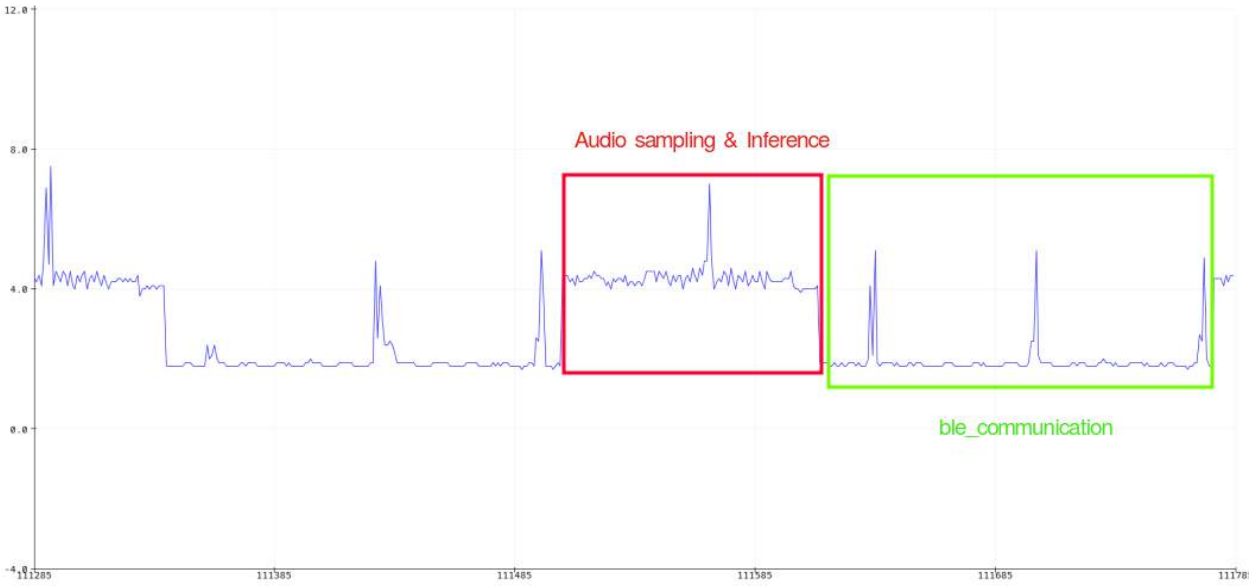
Έτσι για την προσέγγιση της κατανάλωσης ρεύματος χρησιμοποιήθηκε ο ina219 current sensor ,μια μπαταρία λιθίου 190mAh-3.7V για την τροφοδοσία του μικροελεγκτή και το arduino uno για την δειγματοληψία των τιμών του αισθητήρα ρεύματος μέσω I2C.Παρακάτω φαίνεται ο κώδικας που φορτώθηκε από το Arduino Ide στο Uno για τον υπολογισμό των μετρήσεων του ρεύματος κατά την λειτουργία του nrf52840 .

```
#include <Wire.h>
#include <Adafruit_INA219.h>

Adafruit_INA219 ina219;

void setup(void)
{
  Serial.begin(115200);
  while (!Serial) {
      delay(1);
  }

  // Initialize the INA219.
  // By default the initialization will use the largest range (32V, 2A).  However
  // you can call a setCalibration function to change this range (see comments).
  if (! ina219.begin()) {
    Serial.println("Failed to find INA219 chip");
    while (1) { delay(10); }
  }

  //  use a lower 16V, 400mA range (higher precision on volts and amps):
  ina219.setCalibration_16V_400mA();
}

void loop(void)
{
  float current_mA = 0;
  current_mA = ina219.getCurrent_mA();
  Serial.print(current_mA,4);
  Serial.println("");
  delay(1);
}
```

Σύμφωνα με το calibration του ina219 μπορεί να μετρήσει μέγιστο ρεύμα 400mA με βήμα 0.1mA ακρίβεια .Στη συνέχεια το διάγραμμα προκύπτει για το **mfcc-conv1d-56b** μοντέλο ,ωστόσο στην αρχική του μορφή δεν μπορούν να διαχωριστούν τα στάδια επεξεργασίας της πληροφορίας στον μικροελεγκτή .

Εικόνα 5.3 *:Original current diagram for kws app*

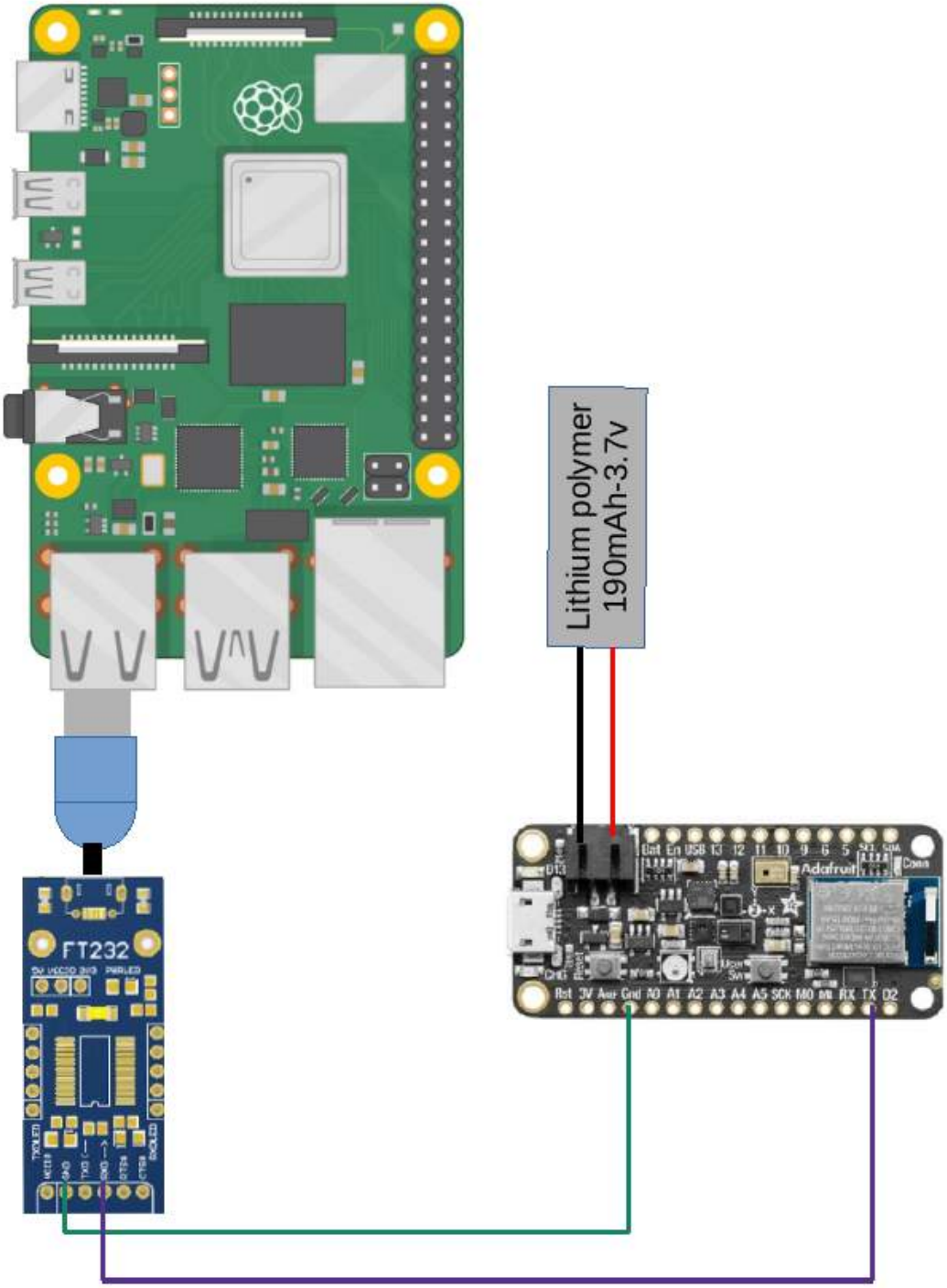
Για αυτό το λόγο εισάγεται κάποια καθυστέρηση μεταξύ των διαφορετικών σταδίων δηλαδή δειγματοληψίας , inference και ble επικοινωνίας ,οπότε προκύπτει το παρακάτω διάγραμμα .



Εικόνα 5.4 *: Modified current diagram*

Επιπλέον για να επιβεβαιώσουμε τις τιμές των διαγραμμάτων στον nrf52840 τρέξαμε την εφαρμογή αναγνώρισης με τροφοδοσία την μπαταρία λιθίου των 190mAh και κράτησε 36 ώρες ,οπότε η μέση κατανάλωση ρεύματος είναι 5.27mA το οποίο δεν απέχει πολύ από το αρχικό διάγραμμα .Ακόμα υπολογίστηκε για το **mfcc-conv1d-941** η μέση κατανάλωση στα 5.93mA και ένας πιθανός λόγος για την αύξηση είναι η μεγαλύτερη πολυπλοκότητα του νευρωνικού του .

Η διάταξη που χρησιμοποιήθηκε για τον υπολογισμό της μέσου ρεύματος ,με την εύρεση της χρονικής διάρκειας για την εξάντληση της μπαταρίας φαίνεται παρακάτω .



Εικόνα 5.5 *:Raspberry pie 4 Measuring Set Up*

```
#include <errno.h>
#include <fcntl.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
int set_interface_attribs (int fd, int speed, int parity)
{
	struct termios tty;
	if (tcgetattr (fd, &tty) != 0)
	{
		printf("error  from tcgetattr");
		 return -1;
	}

	cfsetospeed (&tty, speed);
	cfsetispeed (&tty, speed);

	tty.c_cflag = (tty.c_cflag & ~CSIZE) | CS8;	// 8-bit chars
	// disable IGNBRK for mismatched speed tests; otherwise receive break
	// as \000 chars
	tty.c_iflag &= ~IGNBRK;		// disable break processing
	tty.c_lflag = 0;		// no signaling chars, no echo,
					// no canonical processing
	tty.c_oflag = 0;		// no remapping, no delays
	tty.c_cc[VMIN]  = 0;		// read doesn't block
	tty.c_cc[VTIME] = 5;		// 0.5 seconds read timeout

	tty.c_iflag &= ~(IXON | IXOFF | IXANY); // shut off xon/xoff ctrl

	tty.c_cflag |= (CLOCAL | CREAD);// ignore modem controls,
					// enable reading
	tty.c_cflag &= ~(PARENB | PARODD);	// shut off parity
	tty.c_cflag |= parity;
	tty.c_cflag &= ~CSTOPB;
	tty.c_cflag &= ~CRTSCTS;

	if (tcsetattr (fd, TCSANOW, &tty) != 0)
	{
		printf("error   from tcsetattr");
		return -1;
	}
	return 0;
}

void set_blocking (int fd, int should_block)
{
	struct termios tty;
	memset (&tty, 0, sizeof tty);
	if (tcgetattr (fd, &tty) != 0)
	{
		printf("error   from tggetattr");
		return;
	}

	tty.c_cc[VMIN]  = should_block ? 1 : 0;
```

```
        tty.c_cc[VTIME] = 5;            // 0.5 seconds read timeout

        if (tcsetattr (fd, TCSANOW, &tty) != 0)
                printf("error   setting term attributes");
}

int main(){
    char b[25];
    FILE *fp = popen("ls /dev/ttyUSB0*", "r");
    time_t now;
    char *portname = "/dev/ttyUSB0";
    int fd = open (portname, O_RDWR | O_NOCTTY | O_SYNC);
    if (fd < 0)
{
        printf("error opening port " );
        return 1;
}

    set_interface_attribs (fd, B115200, 0);  // set speed to 115,200 bps, 8n1 (no parity)
    set_blocking (fd, 0);                // set no blocking

    char buf [256];
    fscanf(fp, "%s",b);
    while(strcmp(b, "/dev/ttyUSB0") == 0){

        if (read (fd, buf, sizeof buf)>0){
                time(&now);
                printf("%s ",buf);
                memset(buf, 0, sizeof (buf));
                printf("Today is %s", ctime(&now));
            }
            fscanf(fp, "%s",b);
             }

    return 1;
}
```

Ο παραπάνω κώδικας τρέχει στο raspberry pie 4 και προσθέτει στην έξοδο των αποτελεσμάτων της εφαρμογής την τρέχουσα ημερομηνία και ώρα .Έτσι ώστε να γνωρίζουμε τη συγκεκριμένη χρονική στιγμή που άδειασε η μπαταρία για να υπολογίζουμε με ακρίβεια την μέση κατανάλωση ρεύματος .

6. Μελλοντικές Προοπτικές

Η έρευνα για το σύστημα αναγνώρισης λέξεων απέδειξε ότι είναι δυνατόν όλη η επεξεργασία της πληροφορίας και η εξαγωγή αποτελεσμάτων να γίνεται on-device .Έτσι εύκολα θα μπορούσε να ενσωματωθεί στις λειτουργίες ενός fitness-band ή ενος smart watch ,ώστε για την περιήγηση ο χρήστης εκτός από την οθόνη αφής να μπορεί να χρήστης να λέει κάποιες λέξεις .

Εικόνα 6.1 *: MI BAND 6*

Επιπλέον η αναγνώριση διαφορετικών ήχων και όχι μόνο λέξεων είναι ένα κομμάτι που αξίζει να ερευνήσουμε καθώς υπάρχουν πολλές βιομηχανικές και βιοϊατρικές εφαρμογές, όπως για παράδειγμα συστήματα πρόβλεψης βλάβης μηχανημάτων με αναγνώριση ήχου ή συστήματα αυτόματης διάγνωσης ασθενών .

Η ενσωμάτωση του Machine Learning σε ενσωματωμένες συσκευές γεννάει εφαρμογές που πριν από λίγα χρόνια δεν θα ήταν εφικτές .Παρατηρείται ήδη μια άνοδος της ζήτησης τέτοιων εφαρμογών με παραδείγματα σε wearables όπως στο MI BAND 6 το οποίο διαθέτει stress detection λειτουργία .

Έτσι επιβεβαιώνονται τα λόγια του Pete Warden : ”The Future of ML is Tiny and Bright”.

Βιβλιογραφία

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29(6), 82–97 (2012). https://doi. org/10.1109/MSP.2012.2205597

[2]https://medium.com/@suresh.gopal.k/how-intelligent-voice-response-system-interacts-faba5c05fd1e

[3] https://www.st.com/en/embedded-software/x-cube-ai.html

[4] Geron, A. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems, 2nd edition. O'Reilly Media, Sebastopol, CA, 2019.

[5] Burkov, A. The Hundred-Page Machine Learning Book. Andriy Burkov, 2019.

[6] Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: Keyword spotting on microcontrollers. ArXiv, abs/1711.07128, (2017), 2.

[7] Louis, M. S., Azad, Z., Delshadtehrani, L., Gupta, S., Warden, P., Reddi, V. J., and Joshi, A. Towards deep learning using tensorflow lite on risc-v. Third Workshop on Computer Architecture Research with RISC-V (CARRV), 1, (2019), 6.

[8] Warden P., Why the future of machine learning is tiny. Available on: https://petewarden.com/2018/06/11/why-the-future-of-machinelearning-is-tiny/, [06.07.2020].

[9] Situnayake D., Make deep learning models run fast on embedded hardware. Available on: https://www.edgeimpulse.com/blog/make-deep-learningmodels-run-fast-on-embedded-hardware/, [08.07.2020].

[10] Jongboom J., Introducing EON: neural networks in up to 55less ROM. Available on: https://www.edgeimpulse.com/blog/introducing-eon, [20.11.2020].

[11] Li F., Karpathy A., "Cs231n: Convolutional neural net- works for visual recognition." Stanford University course. Available on: http://cs231n.stanford.edu/, [25.06.2020].

[12] Dive into deep learning, Convolutional Neural Networks. Available on:

http: //d2l.ai/chapter convolutional-neural-networks/conv-layer.html, [17.9.2020].

[13] TensorFlow, GitHub repository. Available on: https://github.com/tensorflow/tensorflow, [21.9.2020].

[14] https://docs.zephyrproject.org/latest/

[15]*https://www.zephyrproject.org/google-and-facebook-select-zephyr-rtos-for-next-generation-products/*

[16] https://github.com/zephyrproject-rtos/zephyr/blob/main/scripts/gen_syscalls.py

[17] https://docs.zephyrproject.org/latest/reference/kernel/scheduling/index.html

[18] https://www.nordicsemi.com/Products/Development-software/nRF-Connect-SDK

[19] https://devzone.nordicsemi.com/

[20]https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/zephyr/getting_started/index.html#install-required-tools

[21] https://learn.adafruit.com/adafruit-feather-sense

[22]https://cdn-learn.adafruit.com/assets/assets/000/049/977/original/MP34DT01-M.pdf

[23] http://www.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

[24] https://www.st.com/resource/en/datasheet/lsm6ds33.pdf

[25]http://adafruit.com/images/product-files/2857/Sensirion_Humidity_SHT3x_Datasheet_digital-767294.pdf

[26]https://cdn-learn.adafruit.com/assets/assets/000/045/848/original/Avago-APDS-9960-datasheet.pdf?1504034182

[27] https://www.st.com/resource/en/datasheet/lis3mdl.pdf

[28]https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Frtc.html

[29]http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[30]https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/examples/micro_speech/train/train_micro_speech_model.ipynb

[31]https://github.com/tensorflow/tensorflow/tree/99fea8da0d98fb271b60b58cfa5755f2bd430079

[32]https://github.com/ShawnHymel/ei-keyword-spotting/blob/master/ei-audio-dataset-curation.ipynb