



ΣΧΟΛΗ Μηχανικής και Τεχνολογίας

**ΤΜΗΜΑ Ηλεκτρολόγων Μηχανικών, Μηχανικών Ηλεκτρονικών Υπολογιστών
και Πληροφορικής**

Πτυχιακή εργασία

Metrodorus OS: System layer for running multiple TinyML models on ESP32-S3

Georgios Athanasiadis

Επιβλέπων Καθηγητής: Dr. Andreas Diavastos

Λεμεσός, 2026

Πνευματικά δικαιώματα

Copyright © Georgios Athanasiadis, 2026.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών, Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Τεχνολογικού Πανεπιστημίου Κύπρου δεν υποδηλώνει απαραίτητως και αποδοχή των απόψεων του/της συγγραφέα εκ μέρους του Τμήματος.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, ο οποίος ήταν πάντα διαθέσιμος να με βοηθήσει όταν αντιμετώπιζα δυσκολίες. Τον ευχαριστώ επίσης για τον χρόνο που αφιέρωσε στις συναντήσεις μας και για τις συμβουλές που μου έδωσε κατά τη διάρκεια της εργασίας. Τέλος θέλω να ευχαριστήσω την οικογένειά μου για τη βοήθεια και την υποστήριξή της, ώστε να μπορέσω να ολοκληρώσω τις σπουδές μου.

Περίληψη

Η παρούσα πτυχιακή εργασία παρουσιάζει το Metrodorus OS, ένα proof of concept system layer για την εκτέλεση πολλών μικρών TinyML μοντέλων σε smartwatch με ESP32-S3. Το βασικό πρόβλημα που εξετάζεται είναι ότι τα περισσότερα embedded AI παραδείγματα είναι κλειστές εφαρμογές, όπου ένα μοντέλο και η λογική του είναι δεμένα μέσα στη firmware. Στόχος της εργασίας ήταν να εξεταστεί αν η firmware μπορεί να λειτουργήσει ως γενική πλατφόρμα για AI packages.

Η υλοποίηση βασίστηκε στο LilyGo T-Watch S3, στο ESP-IDF, στο TensorFlow Lite Micro, στο WAMR και στο LVGL. Τα AI packages αποθηκεύονται στο SPIFFS και αποτελούνται από manifest, TFLite model και WASM module. Με αυτόν τον τρόπο η model-specific λογική του preprocessing και του postprocessing μεταφέρεται στο WASM, ενώ η firmware αναλαμβάνει το hardware, την οθόνη, τον έλεγχο των πακέτων και την εκτέλεση του inference.

Το λειτουργικό proof of concept αντικατέστησε την αρχική firmware του ρολογιού και έδειξε βασική εκτέλεση των packages `sin` και `speech\_recognizer`, μαζί με οθόνες UI και λειτουργική οθόνη AI Market. Το AI Market συνδέεται με Wi-Fi, φορτώνει catalog από GitHub Pages και επιτρέπει την εγκατάσταση διαθέσιμων packages. Τα αποτελέσματα δείχνουν ότι η προσέγγιση είναι εφικτή, αν και χρειάζεται περαιτέρω βελτίωση σε μνήμη, ακρίβεια μοντέλων και διαχείριση πολλών AI λειτουργιών.

Λέξεις κλειδιά: ESP32-S3, TinyML, TensorFlow Lite Micro, WebAssembly, WAMR, smartwatch, AI packages

Abstract

This thesis presents Metrodorus OS, a proof of concept system layer for running multiple small TinyML models on an ESP32-S3 smartwatch. The main problem addressed is that many embedded AI examples are closed applications, where one model and its logic are fixed inside the firmware. The goal of this work was to examine whether the firmware can act as a reusable platform for AI packages.

The implementation was developed for the LilyGo T-Watch S3 using ESP-IDF, TensorFlow Lite Micro, WAMR and LVGL. AI packages are stored in SPIFFS and contain a manifest, a TFLite model and a WASM module. This design moves model-specific preprocessing and postprocessing logic into WASM, while the firmware remains responsible for the hardware, display, package validation and inference execution.

The working proof of concept replaced the original watch firmware and demonstrated the execution of the `sin` and `speech\_recognizer` packages, together with model UI screens and a functional AI Market screen. The AI Market connects to Wi-Fi, loads a catalog from GitHub Pages and allows available packages to be installed. The result shows that this architecture is feasible, although further work is needed in memory optimization, model accuracy and the management of multiple AI functions.

Keywords: ESP32-S3, TinyML, TensorFlow Lite Micro, WebAssembly, WAMR, smartwatch, AI packages

Πίνακας Περιεχομένων

Πνευματικά δικαιώματα	i
Ευχαριστίες	i
Περίληψη	ii
Abstract	iii
Πίνακας Περιεχομένων	iv
Κατάλογος Διαγραμμάτων	vi
Συνοτομογραφίες	vii
1 Εισαγωγή	1
1.1 Πλαίσιο και κίνητρο	1
1.2 Το πρόβλημα των κλειστών παραδειγμάτων TinyML	2
1.3 Γιατί ένα smartwatch χρειάζεται πολλά τοπικά AI μοντέλα	2
1.4 Σκοπός της εργασίας και Metrodorus OS	3
1.5 Δομή της εργασίας	4
1.6 Ονομασία του έργου	4
2 Θεωρητικό Υπόβαθρο	6
2.1 TinyML και embedded systems	6
2.2 TensorFlow Lite Micro	7
2.3 WebAssembly και WAMR	8
2.4 ESP32-S3 και LilyGo T-Watch S3	9
2.5 LVGL και διεπαφή χρήστη	11
3 Ανάλυση και Σχεδιασμός Συστήματος	13
3.1 Βασική ιδέα	13
3.2 Απαιτήσεις σχεδιασμού	14
3.3 Αρχιτεκτονική MultiAI πλατφόρμας	15
3.4 Συμβόλαιο AI package	15
3.5 Διαχείριση lifecycle και ενεργού μοντέλου	16

3.6 Ροή δεδομένων.....	17
3.7 Σχεδιασμός και λειτουργία οθόνης AI Market	18
3.8 Προβλήματα σχεδιασμού και αποφάσεις	18
4 Υλοποίηση	20
4.1 Αρχικό σημείο: TFLM Hello World σε απλή πλακέτα ESP32	20
4.2 Μεταφορά του .tflite στο SPIFFS.....	20
4.3 Πρώτη εναλλαγή μοντέλων με sin/cos και if/else	21
4.4 Ενσωμάτωση WAMR και πρώτο WASM pipeline	22
4.5 Από μεμονωμένο WASM σε δομή AI package.....	23
4.6 Σχεδιασμός ABI μεταξύ WASM και TFLM	24
4.7 Μετάβαση στο LilyGo T-Watch S3 και bring-up hardware.....	25
4.8 Τελική ενοποίηση firmware, UI και lifecycle	26
4.9 Συμπέρασμα της υλοποίησης.....	28
5 Μεθοδολογία Έρευνας και Υλοποίησης.....	29
5.1 Μεθοδολογία ελέγχου μέσω μηνυμάτων κονσόλας	29
5.2 Η νέα εποχή για τον προγραμματισμό: Vibe Coding	30
6 Αποτελέσματα.....	33
6.1 Λειτουργικό proof of concept	33
7 Συμπεράσματα και Μελλοντική Εργασία.....	37
7.1 Συμπεράσματα	37
7.2 Μελλοντική εργασία.....	37
Βιβλιογραφία	38
Παράρτημα I: Παράδειγμα manifest	41

Κατάλογος Διαγραμμάτων

Διάγραμμα 3.1. Βασική αρχιτεκτονική TinyML application.

Διάγραμμα 3.2. Ροή φόρτωσης και inference AI package.

Διάγραμμα 6.1: Κεντρική οθόνη εκκίνησης του Metrodorus OS.

Διάγραμμα 6.2: Οθόνη του AI package `sin` μετά από swipe προς τα δεξιά.

Διάγραμμα 6.3: Οθόνη του `speech\_recognizer`, με αποτέλεσμα `unknown`.

Διάγραμμα 6.4: Οθόνη AI Market με διαθέσιμο package και κατάσταση `Installed`.

Διάγραμμα 6.5: Το ρολόι με κλειστή οθόνη μετά το πάτημα του πλαϊνού κουμπιού.

Συντομογραφίες

Συντομογραφία	Επεξήγηση
AI	Artificial Intelligence
ESP-IDF	Espressif IoT Development Framework
LVGL	Light and Versatile Graphics Library
SPIFFS	SPI Flash File System
TFLM	TensorFlow Lite Micro
WAMR	WebAssembly Micro Runtime
WASM	WebAssembly

1 Εισαγωγή

1.1 Πλαίσιο και κίνητρο

Τα τελευταία χρόνια η τεχνητή νοημοσύνη δεν χρησιμοποιείται μόνο σε μεγάλους servers ή σε ισχυρούς υπολογιστές. Όλο και περισσότερες λειτουργίες AI μεταφέρονται πιο κοντά στον χρήστη, μέσα σε κινητά τηλέφωνα, αισθητήρες, wearables και μικρές embedded συσκευές. Αυτή η κατεύθυνση είναι σημαντική, επειδή πολλές εφαρμογές χρειάζονται γρήγορη απόκριση, μικρή κατανάλωση ενέργειας και καλύτερη προστασία προσωπικών δεδομένων. Όταν το inference γίνεται τοπικά, η συσκευή δεν χρειάζεται κάθε φορά να στέλνει δεδομένα στο cloud. Αυτό μειώνει την καθυστέρηση, λειτουργεί καλύτερα όταν δεν υπάρχει σύνδεση στο διαδίκτυο και περιορίζει τη μεταφορά ευαίσθητων δεδομένων, όπως ήχος, κίνηση ή στοιχεία υγείας [1].

Τα smartwatches είναι ένα πολύ χαρακτηριστικό παράδειγμα αυτής της ανάγκης. Ένα smartwatch βρίσκεται συνεχώς πάνω στο σώμα του χρήστη και μπορεί να διαβάζει δεδομένα από αισθητήρες, όπως accelerometer, gyroscope, μικρόφωνο, heart rate sensor και touch input. Για αυτόν τον λόγο μπορεί να υποστηρίξει μικρά AI μοντέλα για αναγνώριση κίνησης, απλές φωνητικές εντολές, ανίχνευση δραστηριότητας, health-related ειδοποιήσεις ή πιο έξυπνη αλληλεπίδραση με το χρήστη. Στην πράξη, τέτοιες λειτουργίες δεν είναι απλώς θεωρητικές. Η Apple έχει παρουσιάσει on-device Siri σε Apple Watch Series 9 και Apple Watch Ultra 2, ώστε ορισμένα αιτήματα σχετικά με δεδομένα υγείας να μπορούν να επεξεργάζονται στη συσκευή, χωρίς να φεύγουν τα δεδομένα από αυτή [2]. Επίσης, το double tap gesture στο Apple Watch Series 9 βασίζεται σε machine learning algorithm που επεξεργάζεται δεδομένα από accelerometer, gyroscope και optical heart sensor [3].

Υπάρχουν και ερευνητικά παραδείγματα που δείχνουν γιατί το τοπικό AI στα smartwatches είναι χρήσιμο. Το Raise to Speak της Apple περιγράφει έναν low-power detector για voice assistants σε smartwatches, όπου η ανίχνευση γίνεται πάνω στη συσκευή για πιο φυσική ενεργοποίηση του assistant και καλύτερη ιδιωτικότητα [4]. Επίσης, το WatchHAR παρουσιάζει ένα σύστημα Human Activity Recognition που εκτελείται απευθείας σε smartwatch και συνδυάζει δεδομένα ήχου και αδράνειας για αναγνώριση δραστηριοτήτων με χαμηλή καθυστέρηση [5]. Αυτά τα παραδείγματα δείχνουν ότι ένα smartwatch μπορεί να τρέχει περισσότερα από ένα είδη μικρών AI μοντέλων, επειδή οι ανάγκες του χρήστη αλλάζουν ανάλογα με την κατάσταση.

1.2 Το πρόβλημα των κλειστών παραδειγμάτων TinyML

Στον χώρο του TinyML υπάρχουν πολλά παραδείγματα που δείχνουν πώς μπορεί να τρέξει ένα μικρό μοντέλο σε μικροελεγκτή. Αυτά τα παραδείγματα είναι πολύ χρήσιμα για εκμάθηση, όμως συχνά είναι δεμένα με ένα συγκεκριμένο model και με συγκεκριμένο κώδικα μέσα στο firmware. Για παράδειγμα, ένα demo μπορεί να φορτώνει ένα μοντέλο ως C array, να ετοιμάζει το input με σταθερό τρόπο και να εμφανίζει ένα αποτέλεσμα που ταιριάζει μόνο σε εκείνο το μοντέλο. Αυτό είναι αρκετό για ένα πρώτο proof of concept, αλλά δεν είναι αρκετό για μια συσκευή που θέλει να λειτουργήσει σαν μικρή πλατφόρμα.

Το πρόβλημα γίνεται πιο φανερό όταν θέλουμε να προσθέσουμε περισσότερες λειτουργίες. Αν κάθε νέο AI μοντέλο απαιτεί αλλαγή στο firmware, τότε η συσκευή δεν είναι εύκολα επεκτάσιμη. Ο προγραμματιστής πρέπει να ξαναχτίσει το firmware, να ξανακάνει flashing και να προσέχει να μη σπάσει τις υπάρχουσες λειτουργίες. Επιπλέον, το UI μπορεί να αρχίσει να εξαρτάται από λεπτομέρειες κάθε μοντέλου. Αυτό μειώνει τη σαφήνεια της αρχιτεκτονικής και κάνει πιο δύσκολη τη συντήρηση του συστήματος.

Η παρούσα εργασία ξεκινά από την ιδέα ότι το firmware ενός smartwatch δεν πρέπει να είναι απλώς ένα κλειστό demo για ένα μοντέλο. Πρέπει να λειτουργεί ως ένα μικρό runtime περιβάλλον, το οποίο μπορεί να ανακαλύπτει, να ελέγχει και να εκτελεί διαφορετικά AI packages. Κάθε package μπορεί να περιέχει το manifest του, το TFLite model και ένα WASM module που περιγράφει τη model-specific λογική. Με αυτόν τον τρόπο, η συσκευή διαχωρίζει την υποδομή από τη συμπεριφορά του κάθε μοντέλου.

1.3 Γιατί ένα smartwatch χρειάζεται πολλά τοπικά AI μοντέλα

Μια συνηθισμένη ερώτηση είναι γιατί χρειάζεται να εκτελούνται πολλά AI μοντέλα τοπικά σε έναν μικροελεγκτή. Θεωρητικά, θα μπορούσε κάποιος να χρησιμοποιήσει πολλούς μικροελεγκτές, όπου ο καθένας θα τρέχει μία συγκεκριμένη λειτουργία. Θα μπορούσε επίσης να χρησιμοποιήσει ένα πιο ισχυρό SoC, όπως γίνεται σε ακριβότερα smartwatches. Αυτές οι λύσεις όμως δεν είναι πάντα κατάλληλες για ένα μικρό, οικονομικό και ενεργειακά περιορισμένο wearable.

Στο form factor ενός smartwatch ο χώρος είναι πολύ περιορισμένος. Η πλακέτα, η οθόνη, η μπαταρία, οι αισθητήρες, η φόρτιση και το περίβλημα πρέπει όλα να χωρέσουν σε ένα μικρό σώμα που φοριέται στον καρπό. Η χρήση πολλών

μικροελεγκτών αυξάνει την πολυπλοκότητα του hardware, τις γραμμές επικοινωνίας, την κατανάλωση και το κόστος. Ακόμη και αν τεχνικά είναι δυνατό, δεν είναι πρακτικό για μια φθηνή και απλή συσκευή. Η μπαταρία είναι επίσης μικρή, άρα κάθε επιπλέον chip και κάθε επιπλέον active subsystem επηρεάζει τον χρόνο λειτουργίας.

Η επιλογή ενός ισχυρότερου SoC μπορεί να λύσει μέρος του προβλήματος από πλευράς επιδόσεων, αλλά δημιουργεί άλλα προβλήματα. Ένα πιο ισχυρό SoC συνήθως έχει μεγαλύτερο κόστος, πιο σύνθετο software stack, πιθανή ανάγκη για πιο βαρύ λειτουργικό σύστημα και υψηλότερες απαιτήσεις σε μνήμη και ενέργεια. Για μια πτυχιακή εργασία και για ένα proof of concept, ένας μικροελεγκτής όπως ο ESP32-S3 είναι πιο προσιτός, πιο εύκολος στην ανάπτυξη και πιο κοντά στη λογική του TinyML. Δεν προσπαθεί να αντικαταστήσει ένα εμπορικό high-end smartwatch, αλλά να δείξει ότι μπορεί να υπάρξει μια καθαρή και επεκτάσιμη πλατφόρμα AI πάνω σε περιορισμένο hardware.

Ένα ιστορικό παράδειγμα που βοηθά να φανεί αυτή η λογική είναι το Pebble Watch. Το Pebble κινήθηκε σε πιο απλή και οικονομική κατηγορία smartwatch, με έμφαση σε χαμηλή κατανάλωση, πρακτικό σχεδιασμό και μεγάλη διάρκεια μπαταρίας. Ο Eric Migicovsky, απόφοιτος του University of Waterloo στον Καναδά, άρχισε να κατασκευάζει smartwatches όσο σπούδαζε engineering μαζί με φίλους του, και το Pebble έγινε ένα από τα πιο επιτυχημένα Kickstarter projects της εποχής του [6]. Το Wired περιγράφει επίσης ότι το Pebble prototype είχε microcontroller και λίγα megabytes μνήμης, αλλά μπορούσε να προσφέρει ειδοποιήσεις, εφαρμογές και watchfaces [7]. Αυτό δεν σημαίνει ότι κάθε smartwatch πρέπει να έχει την ίδια τεχνική λύση, αλλά δείχνει ότι μια προσεκτική embedded προσέγγιση μπορεί να δώσει χρήσιμη συσκευή χωρίς να βασίζεται απαραίτητα σε πολύ ακριβό hardware.

1.4 Σκοπός της εργασίας και Metrodorus OS

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η σχεδίαση και υλοποίηση ενός proof of concept system layer για την εκτέλεση πολλών TinyML μοντέλων σε ESP32-S3 smartwatch. Το σύστημα ονομάζεται Metrodorus OS. Το όνομα εδώ δεν σημαίνει πλήρες λειτουργικό σύστημα όπως σε έναν υπολογιστή, αλλά ένα μικρό επίπεδο πλατφόρμας πάνω από το firmware. Η βασική ιδέα είναι ότι το firmware αναλαμβάνει τα κοινά μέρη της συσκευής, ενώ η κάθε AI λειτουργία έρχεται ως εξωτερικό package στο filesystem.

Η βασική μορφή ενός AI package στην εργασία είναι: manifest.json, TFLite model και WASM module. Το manifest περιγράφει τα αρχεία και τους βασικούς πόρους που χρειάζεται το package. Το TFLite model περιέχει το νευρωνικό μοντέλο. Το WASM module περιέχει τη λογική που διαφέρει από μοντέλο σε μοντέλο, όπως η προετοιμασία του input, η επεξεργασία του output και το κείμενο που μπορεί να εμφανιστεί στο UI. Έτσι το firmware δεν χρειάζεται να γνωρίζει τις λεπτομέρειες κάθε μοντέλου.

Για την υλοποίηση χρησιμοποιούνται τεχνολογίες όπως ESP-IDF, FreeRTOS, TensorFlow Lite Micro, WAMR, SPIFFS και LVGL. Το θεωρητικό υπόβαθρο και ο ρόλος κάθε τεχνολογίας περιγράφονται πιο αναλυτικά στο επόμενο κεφάλαιο.

Η εργασία δεν έχει ως στόχο να παρουσιάσει ένα τελικό εμπορικό προϊόν. Στόχος είναι να δείξει ότι μπορεί να υπάρξει μια αρχιτεκτονική όπου το smartwatch λειτουργεί σαν μικρή AI πλατφόρμα. Η προσθήκη νέων λειτουργιών δεν πρέπει ιδανικά να απαιτεί αλλαγές στον πυρήνα του firmware. Αντίθετα, το νέο μοντέλο και η model-specific λογική πρέπει να μπαίνουν ως package. Στο proof of concept αυτή η λογική επεκτάθηκε με οθόνη AI Market μέσω Wi-Fi, ώστε διαθέσιμα packages να μπορούν να εμφανίζονται και να εγκαθίστανται χωρίς νέο flashing του firmware.

1.5 Δομή της εργασίας

Η εργασία οργανώνεται σε κεφάλαια που ακολουθούν τη λογική ανάπτυξης του συστήματος. Στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο, δηλαδή οι βασικές έννοιες του TinyML, του TensorFlow Lite Micro, του WebAssembly/WAMR, του ESP32-S3 και του LVGL. Στο Κεφάλαιο 3 γίνεται η ανάλυση και ο σχεδιασμός της αρχιτεκτονικής του Metrodorus OS, με έμφαση στο AI package contract, στη ροή δεδομένων και στο AI Market. Στο Κεφάλαιο 4 παρουσιάζεται η υλοποίηση του firmware, του runtime, του WASM bridge, του AI Manager και της γραφικής διεπαφής. Στο Κεφάλαιο 5 περιγράφεται η μεθοδολογία ανάπτυξης και ελέγχου. Στο Κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα του proof of concept, ενώ στο Κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα και η μελλοντική εργασία.

1.6 Ονομασία του έργου

Το system layer της εργασίας ονομάστηκε Metrodorus OS από τον Μητροδόρο τον Σκήψιο, έναν αρχαίο Έλληνα που έζησε στην αυλή του Ποντίου βασιλιά Μιθριδάτη ΣΤ΄ Ευπάτορα και ήταν γνωστός στην αρχαιότητα για τις μεθόδους βελτίωσης της μνήμης [25]. Η ονομασία επιλέχθηκε συμβολικά, επειδή στο συγκεκριμένο project ένα