ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Προγραμματισμός Ενσωματωμένων Συστημάτων με την MicroPython

ΘΕΟΔΩΡΑΚΟΓΛΟΥ ΕΥΑΓΓΕΛΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΔΑΔΑΛΙΑΡΗΣ ΑΝΤΩΝΙΟΣ
ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

Λαμία ..... έτος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Προγραμματισμός Ενσωματωμένων Συστημάτων με την MicroPython

ΘΕΟΔΩΡΑΚΟΓΛΟΥ ΕΥΑΓΓΕΛΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΔΑΔΑΛΙΑΡΗΣ ΑΝΤΩΝΙΟΣ
ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

Λαμία ..... έτος 2025

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Programming Embedded Systems with MicroPython

THEODORAKOGLOU EVANGELIA

FINAL THESIS

ADVISOR

DADALIARIS ANTONIOS
ASSISTANT PROFESSOR

Lamia ..... year 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 08/07/2025

Ο – Η Δηλ.

ΘΕΟΔΩΡΑΚΟΓΛΟΥ ΕΥΑΓΓΕΛΙΑ

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Τα ενσωματωμένα συστήματα αποτελούν βασικό συστατικό της σύγχρονης τεχνολογίας, ενσωματώνοντας υλικό και λογισμικό για την εκτέλεση εξειδικευμένων λειτουργιών. Χρησιμοποιούνται σε ποικίλες εφαρμογές, από καθημερινές συσκευές έως βιομηχανικούς αυτοματισμούς. Η **MicroPython**, μια ελαφριά έκδοση της **Python**, προσφέρει ευελιξία και απλότητα στον προγραμματισμό τέτοιων συστημάτων, καθιστώντας την ιδανική για πρωτότυπα και εκπαιδευτικούς σκοπούς. Η εξέλιξη των ενσωματωμένων συστημάτων συνεχίζει να επηρεάζει τη τεχνολογία, ενσωματώνοντας καινοτομίες όπως το **IoT[1]** και την τεχνητή νοημοσύνη[2].

ABSTRACT

Embedded systems are a fundamental component of modern technology, integrating hardware and software to perform specialized functions. They are used in a wide range of applications, from everyday devices to industrial automation. MicroPython, a lightweight version of Python, offers flexibility and simplicity in programming such systems, making it ideal for prototyping and educational purposes. The evolution of embedded systems continues to shape technology by incorporating innovations such as the Internet of Things (IoT) and artificial intelligence.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

1.1 Αντικείμενο της εργασίας

Τα ενσωματωμένα συστήματα (**embedded systems**) αποτελούν θεμελιώδες στοιχείο πίσω από πληθώρα τεχνολογικών λύσεων που διαμορφώνουν τον σύγχρονο τρόπο ζωής. Πρόκειται για εξειδικευμένα ψηφιακά συστήματα τα οποία ενσωματώνουν υλικό (**hardware**) και λογισμικό (**software**), σχεδιασμένα να εκτελούν συγκεκριμένες λειτουργίες με υψηλή αποδοτικότητα. Παρότι συχνά <αόρατα> στον τελικό χρήστη, τα ενσωματωμένα συστήματα διαδραματίζουν καθοριστικό ρόλο σε εφαρμογές που κυμαίνονται από φορητές συσκευές έως αυτοματισμούς στη βιομηχανία. [3]

1.2 Στόχοι και σκοπιμότητα της μελέτης

Σκοπός της παρούσας εργασίας είναι η διερεύνηση και ανάλυση των βασικών αρχών των ενσωματωμένων συστημάτων, με ειδική έμφαση στη χρήση της γλώσσας προγραμματισμού **MicroPython**. Η μελέτη καλύπτει δύο κύριους άξονες:

1. Θεωρητικό Υπόβαθρο
    - Ανάλυση των δυνατοτήτων της **MicroPython** σε σύγκριση με άλλες γλώσσες προγραμματισμού
    - Περιγραφή βασικών εννοιών και αρχών σχεδίασης ενσωματωμένων συστημάτων
    - Θεωρητική και πρακτική προσέγγιση της MicroPython

2. Εργαλεία και Προσομοίωση
    - Εξοικείωση με το περιβάλλον **wokwi**
    - Ανάπτυξη και δοκιμή εικονικών πρωτοτύπων

1.3 Ιστορική εξέλιξη

Ο **Charles Stark "Doc" Draper** αναγνωρίζεται παγκοσμίως ως ο πρωτοπόρος που ανέπτυξε το πρώτο πραγματικά λειτουργικό ενσωματωμένο σύστημα. Παράλληλα, διατέλεσε κορυφαίος εμπειρογνώμονας στον τομέα της αδρανειακής πλοήγησης, μια τεχνολογία που βασίζεται σε ειδικούς αισθητήρες όπως επιταχυνσιόμετρα και γυροσκόπια για τον προσδιορισμό της θέσης και της κίνησης. Ως ιδρυτής και διευθυντής του Εργαστηρίου Οργάνωσης του **MIT** (γνωστού σήμερα ως Εργαστήριο **Draper**), ο επιστήμονας αυτός σχεδίασε και υλοποίησε τον επαναστατικό Υπολογιστή Πλοήγησης **Apollo** κατά τη δεκαετία του 1960. Αυτό το σύστημα αποτέλεσε τον εγκέφαλο που διεύθυνε την πορεία κατά τις ιστορικές αποστολές στη Σελήνη, ελέγχοντας με ακρίβεια τόσο την πλοήγηση όσο και τους κρίσιμους ελιγμούς του σκάφους. Το πρώτο σύγχρονο ενσωματωμένο σύστημα ενσωματώθηκε τόσο στη Διαστημική Μονάδα Εντολών όσο και

στη Σεληνιακή Μονάδα του προγράμματος **Apollo**. Ο Υπολογιστής Πλοήγησης **Apollo** αποτέλεσε ένα τεχνολογικό επίτευγμα της εποχής, υποστηρίζοντας την πλοήγηση σε συνολικά εννέα αποστολές προς τη Σελήνη, εκ των οποίων οι έξι κατέληξαν σε επιτυχή προσελήνωση. Κάθε αποστολή διέθετε επιπλέον δύο εξειδικευμένους υπολογιστές: τον **Launch Vehicle Digital Computer (LVDC)**, που συντόνιζε τον πύραυλο κατά την εκτόξευση, και το **Abort Guidance System (AGS)**, το οποίο εξασφάλιζε την ασφαλή απογείωση από τη Σελήνη και την επανένωση με τη μονάδα εντολών. Παρότι οι υπολογιστές αυτοί παρείχαν κρίσιμη υποστήριξη, δεν ήταν σε θέση να υποστηρίξουν την ίδια την προσγείωση στο σεληνιακό έδαφος. Παράλληλα, την ίδια δεκαετία, αναπτύχθηκε και ο υπολογιστής **D-17B**, ο οποίος αποτέλεσε βασικό στοιχείο του συστήματος πλοήγησης των διηπειρωτικών πυραύλων **Minuteman**, επιβεβαιώνοντας τη διείσδυση των ενσωματωμένων συστημάτων σε κρίσιμες εφαρμογές στρατιωτικής τεχνολογίας. Η δεκαετία του 1970 σηματοδότησε μια νέα εποχή για τα ενσωματωμένα συστήματα, καθώς μεταπηδούσαν από στρατιωτικές και διαστημικές εφαρμογές στον εμπορικό τομέα. Η **Volkswagen** άνοιξε το δρόμο το 1969 με την εισαγωγή του πρώτου ενσωματωμένου συστήματος σε αυτοκίνητο, ενώ η **Texas Instruments** και η **Intel** έκαναν επανάσταση το 1971 με την ανάπτυξη του πρώτου μικροελεγκτή και του πρωτοποριακού μονοπύρηνου μικροεπεξεργαστή 4004 αντίστοιχα. Η εξέλιξη των ενσωματωμένων συστημάτων είναι εντυπωσιακή τόσο σε επίπεδο υλικού όσο και λογισμικού. Από τις περίπου 145.000 γραμμές κώδικα που απαιτούνταν για τον Υπολογιστή Πλοήγησης του **Apollo**, φτάσαμε σήμερα σε πολυπλοκότητα εκατομμυρίων γραμμών, με ένα σύγχρονο **smartphone Android** να περιλαμβάνει μεταξύ 12 και 15 εκατομμυρίων γραμμών κώδικα, ενώ σε οχήματα προηγμένης τεχνολογίας ο αριθμός αυτός μπορεί να ξεπερνά τα 100 εκατομμύρια. Η πορεία αυτής της τεχνολογικής ανάπτυξης υπήρξε ραγδαία. Το 1971 εμφανίστηκε ο πρώτος **4-bit** μικροεπεξεργαστής, ενώ ακολούθησε το 1974 ο **8-bit** επεξεργαστής **Intel** 8008 και, την ίδια περίοδο, ο **MC6800** της **Motorola**. Η μετάβαση από τις απλές αριθμομηχανές σε υπερσύγχρονα ενσωματωμένα συστήματα που ελέγχουν κρίσιμες λειτουργίες σε μέσα μεταφοράς, τηλεπικοινωνίες και βιομηχανία, καταδεικνύει πώς η ενσωματωμένη τεχνολογία συνεχίζει να επαναπροσδιορίζει τα όρια του δυνατού.[4]

ΚΕΦΑΛΑΙΟ 2 Βασικές Έννοιες Ενσωματωμένων Συστημάτων

2.1 Τι είναι τα Ενσωματωμένα Συστήματα

Ένα ενσωματωμένο σύστημα (**embedded system**) είναι ένα εξειδικευμένο υπολογιστικό σύστημα σχεδιασμένο να εκτελεί μία προκαθορισμένη λειτουργία, η οποία είναι συγκεκριμένη και αφοσιωμένη σε μία εφαρμογή. Σε αντίθεση με τους υπολογιστές γενικής χρήσης, όπως οι προσωπικοί υπολογιστές (**PC**), τα ενσωματωμένα συστήματα χαρακτηρίζονται από υψηλή εξειδίκευση και συχνά λειτουργούν υπό περιορισμούς πραγματικού χρόνου(**real-time constraints**). Όπως υποδηλώνει και η ονομασία τους συνήθως ενσωματώνονται ως μέρος μεγαλύτερων και πολύπλοκων συστημάτων, αν και σε ορισμένες περιπτώσεις μπορούν να λειτουργήσουν και ως αυτόνομες συσκευές, ανάλογα με τις απαιτήσεις της εφαρμογής. Ο σχεδιασμός των ενσωματωμένων συστημάτων χαρακτηρίζεται από πολυμορφία, καθώς προσαρμόζεται με βάση παράγοντες όπως το μέγεθος, η κατανάλωση ενέργειας, το κόστος και το βάρος. Αυτοί οι παράγοντες καθορίζουν τις τεχνικές προδιαγραφές και τις λύσεις που υλοποιούνται, με στόχο την επίτευξη της βέλτιστης ισορροπίας μεταξύ απόδοσης, αξιοπιστίας και οικονομικής σκοπιμότητας. Σήμερα, τα ενσωματωμένα συστήματα βρίσκονται στον πυρήνα πολλών συσκευών και εφαρμογών, από οικιακές συσκευές και ηλεκτρονικά καταναλωτικά προϊόντα έως βιομηχανικούς αυτοματισμούς, συστήματα μεταφορών και ιατρικό εξοπλισμό, η παρουσία τους είναι καθοριστική για την λειτουργία και την ανάπτυξη της σύγχρονης τεχνολογίας.[5,6]

Εικόνα 1: Ενσωματωμένο σύστημα σε κάρτα με επεξεργαστή, μνήμη, τροφοδοτικό και εξωτερικές διεπαφές (https://shorturl.at/46pso)

2.2 Χαρακτηριστικά και εφαρμογές

Τα ενσωματωμένα συστήματα διακρίνονται από μία σειρά βασικών χαρακτηριστικών, τα οποία τα καθιστούν ιδανικά για συγκεκριμένες εφαρμογές. Αυτά τα χαρακτηριστικά περιλαμβάνουν [8]:

- Εξειδίκευση σε συγκεκριμένες εργασίες:
    - Τα ενσωματωμένα συστήματα δεν είναι πολυλειτουργικά όπως οι γενικού σκοπού υπολογιστές. Αντιθέτως, σχεδιάζονται για να εκτελούν συγκεκριμένες λειτουργίες ή ένα περιορισμένο σύνολο εργασιών, γεγονός που επιτρέπει τη βελτιστοποίηση των επιδόσεων. Χρησιμοποιούνται σε εφαρμογές όπου απαιτείται αυστηρός έλεγχος, συνέπεια και χαμηλή καθυστέρηση απόκρισης.
- Χαμηλό κόστος:
    - Ένα από τα κύρια πλεονεκτήματα των ενσωματωμένων συστημάτων είναι το χαμηλό κόστος παραγωγής και λειτουργίας. Αυτό τα καθιστά οικονομικά προσιτά και κατάλληλα για ευρεία χρήση σε πολλαπλά πεδία.
- Λειτουργία σε πραγματικό χρόνο (**Real-time operation**):
    - Σε πολλές περιπτώσεις, όπως στα συστήματα πλοήγησης, στα ιατρικά μηχανήματα ή στα αμυντικά συστήματα, η απόκριση σε πραγματικό χρόνο είναι ζωτικής σημασίας. Για τον λόγο αυτό, τα ενσωματωμένα συστήματα διαθέτουν λειτουργικά χαρακτηριστικά πραγματικού χρόνου (**RTOS**), που εξασφαλίζουν την έγκαιρη κι αξιόπιστη εκτέλεση εντολών.
- Χαμηλή κατανάλωση ενέργειας:
    - Τα ενσωματωμένα συστήματα σχεδιάζονται για χαμηλή κατανάλωση ενέργειας, κάτι που τα κάνει ιδανικά για φορητές συσκευές και εφαρμογές που λειτουργούν με μπαταρίες.
- Υψηλή απόδοση:
    - Τα ενσωματωμένα συστήματα παρέχουν υψηλή απόδοση στις εργασίες που εκτελούν, χάρη στην εξειδίκευση και την αποδοτική διαχείριση των πόρων.
- Ελάχιστη διεπαφή χρήστη και αυτοματοποίηση:
    - Τα ενσωματωμένα συστήματα συνήθως έχουν ελάχιστη ή απλοποιημένη διεπαφή χρήστη, καθώς σχεδιάζονται για αυτόματη λειτουργία με ελάχιστη ανθρώπινη παρέμβαση, με αποτέλεσμα να αυξάνει την αυτονομία και την αξιοπιστία τους.
- Σταθερότητα και αξιοπιστία:
    - Τα ενσωματωμένα συστήματα είναι εξαιρετικά σταθερά και αξιόπιστα, καθώς σπάνια αλλάζουν ή αναβαθμίζονται. Αυτή η σταθερότητα εξασφαλίζει συνεπή και αξιόπιστη λειτουργία για μεγάλα χρονικά διαστήματα.
- Χρήση μικροεπεξεργαστών και μικροελεγκτών:
    - Τα ενσωματωμένα συστήματα βασίζονται σε μικροεπεξεργαστές ή μικροελεγκτές, οι οποίοι παρέχουν τις απαραίτητες υπολογιστικές δυνατότητες με περιορισμένη χρήση μνήμης και ενέργειας.
- Συμπαγής και οικονομική κατασκευή:

- Η πλειοψηφία των ενσωματωμένων συστημάτων είναι συμπαγής και οικονομικά προσιτά στην κατασκευή. Βασίζονται σε απλό και ελαφρύ υλικό, γεγονός που τα καθιστά εύκολα στην παραγωγή και εγκατάσταση.

Τα ενσωματωμένα συστήματα χρησιμοποιούνται σε ποικίλους τομείς, καθώς συνδυάζουν εξειδίκευση, αποδοτικότητα και αξιοπιστία. Οι κύριες εφαρμογές τους καλύπτουν [7]:

1. Ηλεκτρονικά Καταναλωτικά Προϊόντα
   Τα ενσωματωμένα συστήματα είναι ο πυρήνας πολλών συσκευών που χρησιμοποιούμε καθημερινά, όπως:
   - Κινητά τηλέφωνα και **tablets**: Διαχειρίζονται λειτουργίες επικοινωνίας, πολυμέσων και εφαρμογών.
   - Φορητές συσκευές (**smartwatches, fitness trackers**): Παρακολουθούν δραστηριότητες, μετρήσεις υγείας και παρέχουν ειδοποιήσεις.

2. Οικιακές Συσκευές
   Στο σπίτι, τα ενσωματωμένα συστήματα βρίσκονται σε συσκευές, όπως:
   - Έξυπνες συσκευές (**smart home devices**): Έξυπνες τηλεοράσεις, φωτιστικά, κάμερες ασφαλείας και συστήματα ήχου.
   - Οικιακές συσκευές: Πλυντήρια, ψυγεία, φούρνοι μικροκυμάτων και κλιματιστικά με αυτοματοποιημένες λειτουργίες.

3. Βιομηχανικός Αυτοματισμός
   Στη βιομηχανία, τα ενσωματωμένα συστήματα χρησιμοποιούνται για:
   - Έλεγχος διεργασιών: Αυτοματοποιημένα συστήματα παραγωγής και ελέγχου (**PLC**), συστήματα παρακολούθησης.

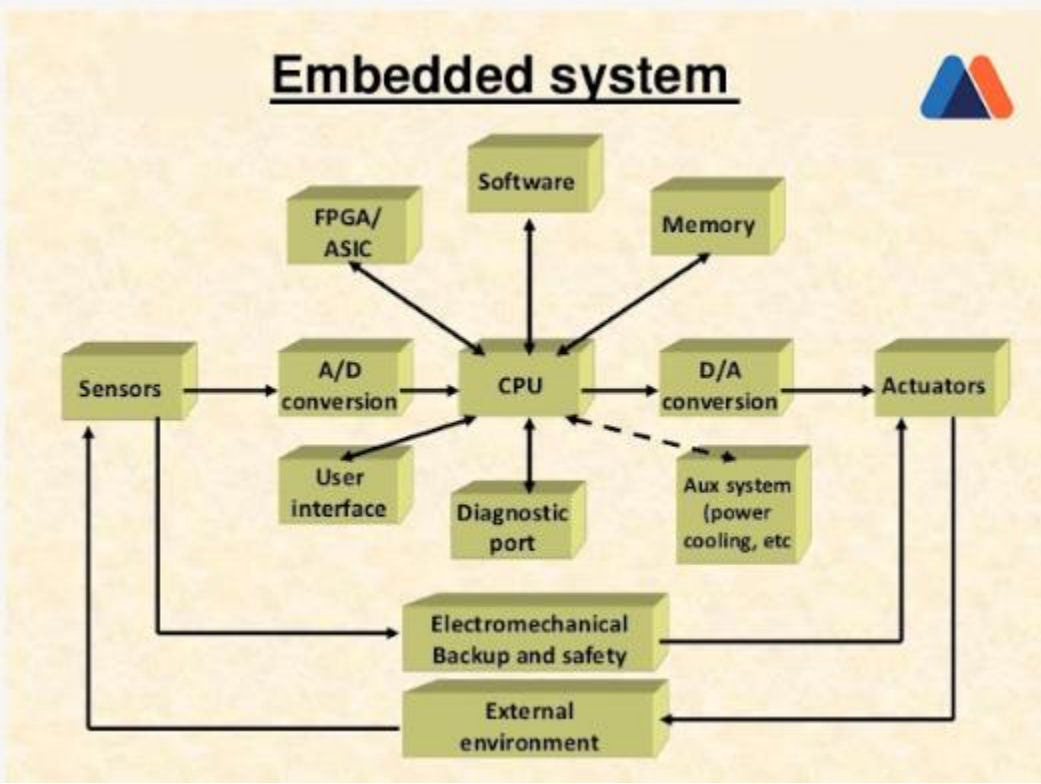
4. Συστήματα Μεταφορών
   Τα ενσωματωμένα συστήματα παίζουν κρίσιμο ρόλο στον τομέα των μεταφορών, όπως:
   - Αυτοκίνητα: Συστήματα πλοήγησης(**GPS**), προειδοποίηση βλάβης, ενεργοποίηση αερόσακου, συστήματα ψυχαγωγίας.
   - Αεροπορία: Αυτόματοι πιλότοι, συστήματα πλοήγησης, συστήματα ασφαλείας σε αεροσκάφη.

5. Ιατρικός Εξοπλισμός
   Στον ιατρικό τομέα, τα ενσωματωμένα συστήματα χρησιμοποιούνται σε:
   - Συσκευές διάγνωσης: Αξονική/Μαγνητική τομογραφία, απεικονιστικές συσκευές(**MRI, CT scanners**).
   - Ιατρικές συσκευές: Παρακολούθηση ζωτικών σημείων, βηματοδότες, μηχανές **CPAP**, μετρητές γλυκόζης.

6. Στρατιωτικές και Αμυντικές Εφαρμογές
   Στην άμυνα, τα ενσωματωμένα συστήματα χρησιμοποιούνται για:
   - Συστήματα πλοήγησης και στόχευσης: Σε **drones**, αεροσκάφη, οχήματα.
   - Συστήματα επικοινωνίας και παρακολούθησης: Ανάλυση δεδομένων, αποστολή πληροφοριών σε πραγματικό χρόνο

2.3 Υλικό και αρχιτεκτονική ενσωματωμένων συστημάτων

Τα ενσωματωμένα συστήματα βασίζονται σε μια ειδικά σχεδιασμένη αρχιτεκτονική που συνδυάζει υλικό (**hardware**) και λογισμικό (**software**) για την εκτέλεση συγκεκριμένων εργασιών.[7]



Εικόνα 2: Αρχιτεκτονική ενσωματωμένου συστήματος (https://www.heavy.ai/technical-glossary/embedded-systems)

1. Βασικά Στοιχεία Υλικού
   - Μικροελεγκτές και Μικροεπεξεργαστής(**Microcontrollers and Microprocessors**):
     - Οι μικροελεγκτές και οι μικροεπεξεργαστές αποτελούν τον εγκέφαλο του συστήματος. Η επιλογή μεταξύ τους, καθώς και η διαφοροποίηση ανάλογα με το bit-width (**8-bit, 16-bit, 32-bit**), καθορίζεται από τις ανάγκες της κάθε εφαρμογής. Οι μικροελεγκτές είναι κατάλληλοι για απλές εφαρμογές, ενώ οι μικροεπεξεργαστές χρησιμοποιούνται σε πιο πολύπλοκες εφαρμογές.

- Μνήμη
    - Η μνήμη στα ενσωματωμένα συστήματα αναλαμβάνει καθοριστικό ρόλο, καθώς επιτρέπει την αποθήκευση και τη διαχείριση δεδομένων και κώδικα που απαιτούνται για τη λειτουργία του συστήματος. Η **RAM** (**Random Access Memory**) προσφέρει γρήγορη πρόσβαση σε προσωρινά δεδομένα, ενώ η **ROM** (**Read-Only Memory**) διασφαλίζει την αποθήκευση μόνιμων πληροφοριών.
- Τροφοδοτικό
    - Η μονάδα τροφοδοσίας είναι υπεύθυνη για την παροχή της απαραίτητης ηλεκτρικής ενέργειας με τάσεις που κυμαίνονται συνήθως από **1,8V** έως **5V**. Η πηγή τροφοδοσίας (ενεργή, μπαταρία, ανεξάρτητη ή κοινή) εξαρτάται από τις ανάγκες της εφαρμογής και την απαιτούμενη αυτονομία.
- Χρονόμετρο και μετρητής
    - Τα χρονόμετρα (**timers**) και οι μετρητές (**counters**) χρησιμοποιούνται για τη διαχείριση χρονικών και αριθμητικών εργασιών. Τα χρονόμετρα επιτρέπουν τη δημιουργία καθυστερήσεων και τον ακριβή χρονισμό, ενώ οι μετρητές χρησιμοποιούνται για την παρακολούθηση συμβάντων.
- Είσοδος/Έξοδος (**Input/Output**)
    - Τα στοιχεία εισόδου και εξόδου επιτρέπουν την αλληλεπίδραση με το εξωτερικό περιβάλλον ώστε να επιτυγχάνετε η παρακολούθηση, ο έλεγχος και η ενημέρωση του χρήστη. Τα στοιχεία εισόδου, όπως οι αισθητήρες, παρέχουν απαραίτητα δεδομένα για την επεξεργασία, ενώ τα στοιχεία εξόδου, όπως οθόνες, κινητήρες, μονάδες επικοινωνίας, μεταδίδουν τα αποτελέσματα της επεξεργασίας.
- Διεπαφή επικοινωνίας
    - Οι διεπαφές επικοινωνίας επιτρέπουν την επικοινωνία μεταξύ συστημάτων και συσκευών. Οι πιο συνηθισμένες τεχνολογίες περιλαμβάνουν **USB, I2C, UART, RS-485, SPI**. Σε απλές εφαρμογές, οι διεπαφές είναι ενσωματωμένες στον μικροελεγκτή, ενώ σε προηγμένες εφαρμογές μπορούν να εγκατασταθούν εξωτερικά για μεγαλύτερη ευελιξία.
- Ηλεκτρικό κύκλωμα
    - Τα ηλεκτρικά κυκλώματα συνδέουν τα διάφορα στοιχεία του συστήματος. Περιλαμβάνουν εξαρτήματα όπως αντιστάσεις, πυκνωτές, δίοδους, τρανζίστορ και επαγωγείς, τα οποία εξασφαλίζουν τη σωστή λειτουργία του συστήματος.
- Πλακέτα τυπωμένου κυκλώματος (**PCB – Printed Circuit Boards**)
    - Με τη χρήση αγώγιμων ιχνών χαλκού, οι PCBs επιτρέπουν τη δημιουργία πολύπλοκων κυκλωμάτων σε μικρό χώρο, ενώ προσφέρουν ευκολία στη συντήρηση και την παραγωγή.
- Αντίσταση
    - Οι αντιστάσεις (**resistors**) χρησιμοποιούνται για τον έλεγχο της ροής του ρεύματος και τη ρύθμιση των επιπέδων σήματος. Υπάρχουν δύο κύριοι τύποι: σταθερές, που έχουν μια καθορισμένη τιμή αντίστασης και

μεταβλητές, που μπορούν να χρησιμοποιηθούν ως αισθητήρες ή για ρύθμιση.

- Πυκνότητα
    - Οι πυκνωτές παρέχουν τη δυνατότητα αποθήκευσης και απελευθέρωσης ενέργειας, όταν το κύκλωμα το απαιτεί. Χρησιμοποιούνται σε εφαρμογές, όπως η εξομάλυνση τάσης, η παράκαμψη θορύβου, φιλτράρισμα σημάτων.
- Δίοδος
    - Η δίοδος επιτρέπει τη ροή του ρεύματος μόνο προς μία κατεύθυνση. Κατασκευασμένη από ημιαγωγικά υλικά όπως πυρίτιο ή γερμάνιο, η δίοδος χρησιμοποιείται σε εφαρμογές όπως ρυθμιστές τάσης, λογικές πύλες, κυκλώματα επεξεργασίας σημάτων.
- Τρανζίστορ
    - Τα τρανζίστορ παρέχουν τις λειτουργίες εναλλαγής και ενίσχυσης. Οι δύο κύριοι τύποι είναι: τα **MOSFET** (ελεγχόμενα τάσης) και τα BJT (ελεγχόμενα ρεύματος).
- Ολοκληρωμένο κύκλωμα (**ICs – Intergrated Circuits**)
    - Τα ολοκληρωμένα κυκλώματα (**ICs**) συνδυάζουν πολλά ηλεκτρικά εξαρτήματα σε ένα ενιαίο τσιπ. Τα **ICs** μπορούν να λειτουργήσουν ως ταλαντωτές, μικροεπεξεργαστές, ενισχυτές, μονάδες μνήμης κ.α.
- Δίοδος εκπομπής φωτός (**LED**)
    - Τα **LEDs** παρέχουν οπτική ένδειξη, υποδεικνύοντας την κατάσταση του κυκλώματος και ειδοποιώντας για σφάλματα (άναμμα, σβήσιμο, αναβοσβήσιμο).
- Επαγωγέας
    - Ο επαγωγέας είναι ένα βασικό ηλεκτρικό εξάρτημα που χρησιμοποιείται για την αποθήκευση ενέργειας σε ένα μαγνητικό πεδίο όταν διέρχεται ηλεκτρικό ρεύμα από αυτόν. Οι επαγωγείς έχουν την ιδιότητα να εμποδίζουν τις μεταβολές του ρεύματος επιτρέποντας τη ροή συνεχούς ρεύματος ενώ αποκλείουν το εναλλασσόμενο ρεύμα.

2. Βασικά Στοιχεία Λογισμικού
    - Λειτουργικό Σύστημα (**RTOS – Real-Time Operating System**)
        - Τα λειτουργικά συστήματα πραγματικού χρόνου έχουν ως κύριο στόχο την έγκαιρη εκτέλεση κρίσιμων εργασιών, εξασφαλίζοντας ότι οι διεργασίες ολοκληρώνονται εντός αυστηρών χρονικών ορίων. Χάρη στην ικανότητα τους να διαχειρίζονται πολλαπλές εργασίες ταυτόχρονα και στην υψηλή αξιοπιστία τους, τα **RTOS** είναι ιδανικά για εφαρμογές όπου ο χρονισμός είναι κρίσιμος.[8]

    - Επεξεργαστής κειμένου
        - Ο επεξεργαστής κειμένου χρησιμοποιείται για τη σύνταξη και την αποθήκευση του πηγαίου κώδικα σε γλώσσες προγραμματισμού όπως **C** και **C++**, αποθηκεύοντάς τον ως αρχείο κειμένου.

- Μεταγλωττιστής
    - Ο μεταγλωττιστής μεταφράζει τον πηγαίο κώδικα σε γλώσσα μηχανής χαμηλού επιπέδου, όπως κώδικας μηχανής, γλώσσα συναρμολόγησης, κώδικας αντικειμένου. Αυτή η διαδικασία είναι απαραίτητη για τη δημιουργία εκτελέσιμων αρχείων, τα οποία μπορούν να εκτελεστούν από το υλικό του ενσωματωμένου συστήματος.

- Συμβολομεταφραστής
    - Ανάλογα με τη γλώσσα προγραμματισμού τα ενσωματωμένα συστήματα βασίζονται σε **assembler** είτε σε μεταγλωττιστή. Ο **assembler** μεταφράζει τον κώδικα **assembly** σε κώδικα αντικειμένου και μετά σε κώδικα **HEX**, ο οποίος γράφεται στο τσιπ. Ο μεταγλωττιστής μεταφράζει τον κώδικα υψηλού επιπέδου απευθείας σε γλώσσα μηχανής.

- Εξομοιωτής
    - Ο εξομοιωτής προσομοιώνει τη λειτουργία του συστήματος σε πραγματικό χρόνο, επιτρέποντας στον έλεγχο του κώδικα, διόρθωση σφαλμάτων χωρίς την πραγματική εκτέλεση της λειτουργίας.

- Επεξεργαστής συνδέσμων
    - Ο επεξεργαστής συνδέσμων συνδέει πολλά αρχεία αντικειμένων και τα συνδυάζει σε ένα ενιαίο εκτελέσιμο αρχείο. Αυτή η διαδικασία είναι απαραίτητη για την οργάνωση του κώδικα.

- Εντοπισμός σφαλμάτων
    - Το πρόγραμμα εντοπισμού σφαλμάτων χρησιμοποιείται για τον εντοπισμό, ανάλυση και διόρθωση σφαλμάτων στον κώδικα. Επιτρέπει στους προγραμματιστές να παρακολουθούν τις τιμές των μεταβλητών και να ορίζουν σημεία διακοπής.

ΚΕΦΑΛΑΙΟ 3 Γλώσσες Προγραμματισμού για Ενσωματωμένα Συστήματα

3.1 Παραδοσιακές γλώσσες (Assembly, C, C++)

1. **Assembly**

Η γλώσσα **Assembly** αποτελεί έναν από τους ακρογωνιαίους λίθους του προγραμματισμού χαμηλού επιπέδου, διατηρώντας μια θεμελιώδη θέση στην αρχιτεκτονική των σύγχρονων υπολογιστικών συστημάτων. Ως μία από τις πρώτες γλώσσες προγραμματισμού που αναπτύχθηκαν, συνεχίζει να διαδραματίζει κρίσιμο ρόλο στην επιστήμη των υπολογιστών, παρέχοντας άμεσο έλεγχο και πρόσβαση στους μηχανισμούς λειτουργίας του υλικού. Αντί για τις αφαιρετικές δομές των γλωσσών υψηλού επιπέδου, όπως η **C++** ή η **Python**, η **Assembly** χρησιμοποιεί συμβολικές εντολές, γνωστές ως μνημονικούς κωδικούς, οι οποίες αντιστοιχούν άμεσα στις εντολές μηχανής. Λειτουργεί ως ενδιάμεσο στάδιο μεταξύ της δυαδικής γλώσσας, που κατανοεί ο επεξεργαστής, και των γλωσσών υψηλού επιπέδου, που είναι πιο εύχρηστες για τους ανθρώπους. Η ιστορία της **Assembly** ξεκινά τη δεκαετία του 1940, όταν οι πρώτοι υπολογιστές λειτουργούσαν με λυχνίες κενού και ο προγραμματισμός γινόταν απευθείας σε δυαδική μορφή. Η εμφάνιση της **Assembly** αποτέλεσε σταθμό, καθώς εισήγαγε πιο κατανοητούς συμβολισμούς για την περιγραφή των εντολών του υπολογιστή. Με την ανάπτυξη των τρανζίστορ τη δεκαετία του 1950, οι υπολογιστές έγιναν πιο σταθεροί και ικανοί, γεγονός που οδήγησε στη δημιουργία πιο σύνθετων μορφών **Assembly** και στην εμφάνιση των πρώτων γλωσσών υψηλού επιπέδου, όπως η **FORTRAN**. Τη δεκαετία του 1960, με την υιοθέτηση των ολοκληρωμένων κυκλωμάτων, η **Assembly** εμπλουτίστηκε με νέα χαρακτηριστικά, όπως μακροεντολές και συμβολικές ετικέτες, που βελτίωσαν την αναγνωσιμότητα και την αποδοτικότητα του κώδικα. Η δεκαετία του 1970 σηματοδότησε την επανάσταση των μικροεπεξεργαστών και των προσωπικών υπολογιστών, καθιστώντας την **Assembly** προσβάσιμη σε περισσότερους προγραμματιστές. Από τη δεκαετία του 1980 και εξής, η **Assembly** προσαρμόστηκε στις απαιτήσεις των πολυπύρηνων συστημάτων και της παράλληλης επεξεργασίας, υποστηρίζοντας ολοένα και πιο πολύπλοκα συστήματα με εξειδικευμένα εργαλεία ανάλυσης και αποσφαλμάτωσης. Η **Assembly** βασίζεται στη χρήση μνημονικών εντολών, οι οποίες μετατρέπονται σε γλώσσα μηχανής με τη βοήθεια ενός προγράμματος που ονομάζεται **assembler**. Ο κώδικας που γράφει ο προγραμματιστής αποθηκεύεται σε αρχείο με κατάληξη όπως **.asm** και στη συνέχεια μετατρέπεται σε εκτελέσιμη μορφή μέσω των σταδίων της μετάφρασης, της δημιουργίας αντικειμενικών αρχείων και του **linking**. Σημαντικά στοιχεία της **Assembly** είναι οι καταχωρητές, όπως **AX**, **BX** και **CX**, που αποτελούν μικρές μονάδες μνήμης στον επεξεργαστή για προσωρινή αποθήκευση τιμών, καθώς και οι εντολές όπως **ADD** (addition-πρόσθεση), **MOV** (move-αντιγραφή) και **CMP** (compare-σύγκριση), που καθοδηγούν τον επεξεργαστή να εκτελέσει συγκεκριμένες πράξεις. Οι μακροεντολές επιτρέπουν την επαναχρησιμοποίηση κώδικα, ενώ τα **labels** χρησιμεύουν για τον ορισμό σημείων στον κώδικα με συμβολικά ονόματα. Οι

τελεστέοι (**operands**) είναι οι τιμές ή οι διευθύνσεις στις οποίες εφαρμόζονται οι εντολές. Ο ίδιος ο κώδικας περιλαμβάνει τα λεγόμενα **opcodes**, δηλαδή τους μνημονικούς κωδικούς εντολών. Η **Assembly** συνδέεται στενά με το δυαδικό σύστημα, που αποτελεί τη βάση όλων των υπολογιστικών λειτουργιών. Το δυαδικό σύστημα χρησιμοποιεί μόνο τα ψηφία 0 και 1. Για παράδειγμα, ο αριθμός 45 εκφράζεται ως 101101, δηλαδή 32 + 8 + 4 + 1. Εκτός από το δυαδικό, χρησιμοποιείται και το δεκαεξαδικό σύστημα, που βασίζεται στο 16 και χρησιμοποιεί τα ψηφία 0-9 και τα γράμματα A-F. Για τη μετατροπή ενός δεκαδικού αριθμού σε δεκαεξαδικό, διαιρούμε τον αριθμό με το 16 και καταγράφουμε τα υπόλοιπα. Για παράδειγμα, το 450 γίνεται 1C2, καθώς 450/16 = 28 υπόλοιπο 2, 28/16 = 1 υπόλοιπο 12 (δηλαδή C), και 1/16 = 0 υπόλοιπο 1. Η αντίστροφη μετατροπή, από δεκαεξαδικό σε δεκαδικό, γίνεται με πολλαπλασιασμό κάθε ψηφίου με κατάλληλη δύναμη του 16. Έτσι, το A7B ισούται με 10×256 + 7×16 + 11 = 2683. Η διαδικασία ανάπτυξης προγραμμάτων σε **Assembly** ξεκινά με τη συγγραφή κώδικα χρησιμοποιώντας μνημονικά σύμβολα σε έναν απλό επεξεργαστή κειμένου. Ο **assembler** μεταφράζει αυτά τα σύμβολα σε δυαδικές εντολές που μπορεί να εκτελέσει άμεσα ο επεξεργαστής. Αξίζει να σημειωθεί ότι η **Assembly** είναι στενά εξαρτώμενη από την αρχιτεκτονική του επεξεργαστή. Υπάρχουν διαφορετικές εκδόσεις της γλώσσας για διαφορετικές οικογένειες επεξεργαστών, όπως **x86**, **ARM** και **MIPS**, με αποτέλεσμα ο κώδικας να μην είναι φορητός μεταξύ διαφορετικών συστημάτων. Αυτή η εξάρτηση από το υλικό, σε συνδυασμό με την απαιτητική διαδικασία αποσφαλμάτωσης και ανάπτυξης, συγκαταλέγεται στα βασικά μειονεκτήματα της γλώσσας. Παρόλα αυτά, η **Assembly** προσφέρει ξεχωριστό έλεγχο στο υλικό του υπολογιστή και οδηγεί σε εξαιρετικά αποδοτικό και βελτιστοποιημένο κώδικα. Είναι ιδανική για εφαρμογές που απαιτούν άμεση και λεπτομερή διαχείριση των πόρων, όπως οι μικροελεγκτές, οι αισθητήρες και τα ενσωματωμένα συστήματα. Η γλώσσα αυτή διαδραματίζει σημαντικό ρόλο στην ασφάλεια πληροφοριακών συστημάτων, στην ανάλυση κακόβουλου λογισμικού και στον εντοπισμό ευπαθειών. Επίσης, είναι απαραίτητη στην ανάπτυξη λειτουργικών συστημάτων, οδηγών συσκευών και του πυρήνα (**kernel**), όπου η μέγιστη απόδοση και ο ακριβής έλεγχος είναι ζωτικής σημασίας. Αν και οι σύγχρονοι μεταγλωττιστές είναι ικανοί να παράγουν πολύ αποδοτικό κώδικα μηχανής, σε συγκεκριμένες περιπτώσεις ένας έμπειρος προγραμματιστής **Assembly** μπορεί να επιτύχει ακόμα μεγαλύτερη απόδοση. Επιπλέον, η κατανόηση της **Assembly** παρέχει βαθιά γνώση του τρόπου λειτουργίας των υπολογιστών, προσφέροντας πολύτιμες γνώσεις ακόμα και για προγραμματιστές που εργάζονται κυρίως με γλώσσες υψηλού επιπέδου. Η εκμάθηση της **Assembly** απαιτεί σημαντική προσπάθεια και δέσμευση, καθώς προϋποθέτει κατανόηση των βασικών αρχών οργάνωσης και αρχιτεκτονικής υπολογιστών, όπως το μοντέλο μνήμης, η λειτουργία των καταχωρητών και η εκτέλεση εντολών. Ωστόσο, αυτή η επένδυση μπορεί να αποδειχθεί εξαιρετικά επωφελής, παρέχοντας μια θεμελιώδη κατανόηση που βελτιώνει τις δεξιότητες προγραμματισμού σε όλα τα επίπεδα. Συμπερασματικά, η γλώσσα **Assembly**, παρά την ηλικία της και την εμφάνιση πιο προσιτών εναλλακτικών, συνεχίζει να διαδραματίζει καίριο ρόλο στον τομέα της προφορικής. Η άμεση σύνδεση της με υλικό του υπολογιστή την καθιστά αναντικατάστατο εργαλείο για εφαρμογές που απαιτούν υψηλή απόδοση, μειωμένη κατανάλωση πόρων και πλήρη έλεγχο της συμπεριφοράς του συστήματος.[9-13]

2. C

Η γλώσσα προγραμματισμού **C** αποτελεί μία από τις πιο σημαντικές και ευρέως χρησιμοποιούμενες γλώσσες γενικού σκοπού, με εκτεταμένες εφαρμογές στον τομέα της πληροφορικής. Δημιουργήθηκε στα **Bell Labs** το 1972 από τους **Dennis Ritchie** και **Ken Thompson**, οι οποίοι εργάζονταν πάνω στο λειτουργικό σύστημα **Unix**. Επειδή η γλώσσα **Assembly** ήταν περιοριστική για τη δημιουργία βοηθητικών εργαλείων, ανέπτυξαν τη **C** ώστε να είναι πιο ευέλικτη και αποδοτική, με δυνατότητα άμεσης απεικόνισης σε εντολές μηχανής. Η **C** εξελίχθηκε μέσα από τις δεκαετίες, αρχικά με την έκδοση **K&R C** το 1978, στη συνέχεια με την καθιέρωση του προτύπου **ANSI C** το 1989, και αργότερα με τις αναβαθμισμένες εκδόσεις **C99**, **C11** και **C18**, οι οποίες εισήγαγαν νέες λειτουργίες, βελτιώσεις και τεχνικές διευκρινίσεις. Η κληρονομία της **C** στον κόσμο του προγραμματισμού είναι αδιαμφισβήτητη. Έχει διαμορφώσει καθοριστικά τη σύνταξη και τη φιλοσοφία πολλών σύγχρονων γλωσσών προγραμματισμού. Η **C++**, που αναπτύχθηκε ως επέκταση της **C**, προσέθεσε αντικειμενοστραφή χαρακτηριστικά διατηρώντας παράλληλα τη συμβατότητα με την προκάτοχό της. Η **Java** υιοθέτησε πολλά στοιχεία της σύνταξης της **C**, ενώ πρόσθεσε αυτόματη διαχείριση μνήμης και εικονική μηχανή για διαπλατφορμική λειτουργία. Η **Python**, αν και διαφέρει σημαντικά σε φιλοσοφία, έχει επίσης επηρεαστεί από τη **C**, με πολλές από τις βασικές της βιβλιοθήκες να είναι υλοποιημένες σε αυτήν. Η **Go**, μια πιο πρόσφατη γλώσσα από την **Google**, επίσης αντλεί έμπνευση από τη σύνταξη και το μοντέλο εκτέλεσης της **C**. Αυτή η ευρεία επιρροή έχει καθιερώσει τη **C** ως τη "μητρική γλώσσα" του σύγχρονου προγραμματισμού. Η **Embedded C** είναι μια ειδική επέκταση της γλώσσας **C**, σχεδιασμένη για να ανταποκρίνεται στις ανάγκες των ενσωματωμένων συστημάτων, όπως μικροελεγκτές και ηλεκτρονικές συσκευές με περιορισμένους πόρους. Δημοσιεύτηκε πρώτη φορά ως τεχνική αναφορά το 2004 και αναθεωρήθηκε το 2016. Ως γλώσσα χαμηλού επιπέδου, προσφέρει στους προγραμματιστές μεγαλύτερο έλεγχο σε κρίσιμους τομείς όπως η διαχείριση μνήμης και η κατανάλωση ενέργειας, που είναι καθοριστικοί παράγοντες για συστήματα με περιορισμένες δυνατότητες. Επιπλέον, η **Embedded C** περιλαμβάνει εξειδικευμένες λειτουργίες για είσοδο/έξοδο, αριθμητική σταθερού σημείου, πρόσβαση σε διευθύνσεις υλικού και διαχωρισμό της μνήμης σε πολλαπλές περιοχές. Οι βασικοί τύποι δεδομένων στη γλώσσα **Embedded C** περιλαμβάνουν συναρτήσεις, ακέραιους, **bits** και πραγματικούς αριθμούς. Οι συναρτήσεις μπορούν να επιστρέφουν τιμές ή να μην επιστρέφουν τίποτα (μέσω της λέξης-κλειδί **void**) και μπορεί να δέχονται ή όχι παραμέτρους. Οι ακέραιοι διαχωρίζονται σε τύπους όπως **int**, **short** και **long**, με δυνατότητα επιλογής του κατάλληλου μήκους για εξοικονόμηση μνήμης. Για τον χειρισμό μεμονωμένων **bits**, χρησιμοποιούνται οι τύποι **bit** και **bits**, οι οποίοι επιτρέπουν τον άμεσο χειρισμό μονάδων πληροφορίας. Οι πραγματικοί αριθμοί, δηλαδή τα δεκαδικά, αποφεύγονται στις περισσότερες ενσωματωμένες εφαρμογές λόγω του υψηλού κόστους σε υπολογιστική ισχύ και μνήμη. Πέρα από αυτούς, η **Embedded C** υποστηρίζει και πιο σύνθετους τύπους όπως πίνακες, δείκτες, απαριθμήσεις, δομές και ενώσεις. Η βασική δομή ενός προγράμματος σε **Embedded C** περιλαμβάνει διάφορα στάδια. Αρχικά καταγράφεται τεκμηρίωση (**documentation**), η οποία περιγράφει πληροφορίες όπως το όνομα του αρχείου, ο συγγραφέας και η ημερομηνία δημιουργίας. Στη συνέχεια, προστίθενται οδηγίες προεπεξεργασίας που ξεκινούν με το σύμβολο # και

έχουν σκοπό να καθορίσουν πώς θα μεταγλωττιστεί το πρόγραμμα, εισάγοντας βιβλιοθήκες ή καθορίζοντας συνθήκες μεταγλώττισης. Ακολουθεί η δήλωση των καθολικών μεταβλητών, οι οποίες μπορούν να χρησιμοποιηθούν σε όλο το πρόγραμμα. Το κύριο μέρος του προγράμματος αρχίζει με τη συνάρτηση **main**, η οποία αποτελεί το σημείο εκκίνησης της εκτέλεσης. Εντός της **main** γίνονται δηλώσεις τοπικών μεταβλητών, αρχικοποιήσεις θυρών ή συσκευών, και στη συνέχεια ακολουθεί το κύριο σώμα του προγράμματος, δηλαδή η υλοποίηση της λογικής που αφορά την εκάστοτε εφαρμογή. Τέλος, μπορεί να υπάρχουν επιπλέον υποπρογράμματα, δηλαδή άλλες συναρτήσεις που καλούνται από τη main για την εκτέλεση επιμέρους εργασιών. Ένα χαρακτηριστικό παράδειγμα **Embedded C** φαίνεται σε έναν απλό κώδικα που ελέγχει την ενεργοποίηση ενός **LED** μέσω ενός πλήκτρου. Αρχικά εισάγονται οι βιβλιοθήκες και δηλώνονται οι θύρες που χρησιμοποιούνται για ανάγνωση και εγγραφή. Δηλώνονται μερικές σταθερές τιμές όπως **ON**, **OFF** και **PUSHED**. Η συνάρτηση main αρχικοποιεί τις θύρες, ορίζοντας ποια **pins** είναι είσοδοι και ποια έξοδοι. Μέσα σε ένα ατέρμονο βρόχο (**while**(1)), γίνεται έλεγχος αν το κουμπί έχει πατηθεί. Αν ναι, καλείται μια υποθετική συνάρτηση αναμονής (**wait**), και στη συνέχεια, αν εξακολουθεί να είναι πατημένο, ενεργοποιείται το **LED**, παραμένει ανοιχτό για λίγα δευτερόλεπτα και μετά σβήνει. Με αυτόν τον τρόπο υλοποιείται μια βασική αλληλεπίδραση χρήστη-συστήματος με χρήση της γλώσσας **Embedded C**. Συνοψίζοντας, η **Embedded C** είναι μια αναπόσπαστη τεχνολογία στον τομέα των ενσωματωμένων συστημάτων, επιτρέποντας στους μηχανικούς να ελέγχουν άμεσα τον υλικό εξοπλισμό και να προσαρμόζουν τον προγραμματισμό τους στις απαιτήσεις του εκάστοτε μικροελεγκτή. Η κατανόηση των βασικών δομών και της λειτουργίας της γλώσσας αποτελεί θεμέλιο λίθο για όποιον θέλει να ασχοληθεί σοβαρά με τον σχεδιασμό και την ανάπτυξη έξυπνων ηλεκτρονικών συσκευών. Στον σημερινό τεχνολογικό τοπίο, η **C** συνεχίζει να διαδραματίζει σημαντικό ρόλο στην ανάπτυξη λογισμικού, παρά την εμφάνιση νεότερων γλωσσών με πιο σύγχρονα χαρακτηριστικά. Η εκμάθησή της παραμένει θεμελιώδης για τους επίδοξους προγραμματιστές, καθώς παρέχει κρίσιμες γνώσεις για την κατανόηση των βασικών αρχών της επιστήμης των υπολογιστών και τη λειτουργία των υπολογιστικών συστημάτων σε χαμηλό επίπεδο. Συμπερασματικά, η γλώσσα **C** αποτελεί έναν αναντικατάστατο πυλώνα του σύγχρονου προγραμματισμού, συνδυάζοντας αποτελεσματικά τη δύναμη του προγραμματισμού χαμηλού επιπέδου με την εκφραστικότητα μιας δομημένης γλώσσας. Η διαχρονική της αξία και η συνεχιζόμενη επιρροή της στην ανάπτυξη νέων γλωσσών και τεχνολογιών μαρτυρούν την εξαιρετική σχεδίαση και λειτουργικότητά της, καθιστώντας την έναν ακρογωνιαίο λίθο στην εξέλιξη της επιστήμης των υπολογιστών.[15,16]

3. C++

Η γλώσσα **C++** δημιουργήθηκε από τον **Bjarne Stroustrup** το 1979 στα **Bell Labs**, αρχικά με την ονομασία "**C with Classes**", προτού εξελιχθεί στη μορφή που γνωρίζουμε σήμερα. Στον πυρήνα της ανάπτυξης των ενσωματωμένων συστημάτων βρίσκεται η γλώσσα προγραμματισμού **C++**, η οποία έχει αποκτήσει σημαντική απήχηση λόγω των μοναδικών πλεονεκτημάτων που προσφέρει. Η **C++** προσφέρει έναν ιδανικό συνδυασμό υψηλού και χαμηλού επιπέδου προγραμματισμού, επιτρέποντας στους μηχανικούς να γράφουν κώδικα που είναι ταυτόχρονα

αποδοτικός και εύκολα συντηρήσιμος. Ένα από τα κύρια χαρακτηριστικά που καθιστά την **C++** ιδανική για τέτοιες εφαρμογές είναι η ικανότητά της για αποδοτική διαχείριση πόρων. Τα ενσωματωμένα συστήματα συχνά λειτουργούν σε περιβάλλοντα με περιορισμένη υπολογιστική ισχύ, μικρή διαθέσιμη μνήμη και ενέργεια, και η **C++** επιτρέπει στους προγραμματιστές να ελέγχουν με ακρίβεια τη χρήση μνήμης και την απόδοση του συστήματος. Η αντικειμενοστραφής φύση της **C++** προσφέρει σημαντικά πλεονεκτήματα στον σχεδιασμό και την ανάπτυξη ενσωματωμένων συστημάτων. Μέσω της χρήσης κλάσεων, αφαιρέσεων και ενθυλάκωσης, οι μηχανικοί μπορούν να δημιουργήσουν καλύτερα οργανωμένο, επαναχρησιμοποιήσιμο και ευέλικτο κώδικα. Αυτό είναι ιδιαίτερα σημαντικό σε πολύπλοκα συστήματα όπου η συντήρηση και η επέκταση του λογισμικού παίζουν κρίσιμο ρόλο. Επιπλέον, η φορητότητα του κώδικα **C++** είναι ένα ακόμη σημαντικό πλεονέκτημα. Δεδομένου ότι τα ενσωματωμένα συστήματα αναπτύσσονται για ποικίλες πλατφόρμες και αρχιτεκτονικές, η δυνατότητα μεταγλώττισης του ίδιου κώδικα σε διαφορετικά υλικολογισμικά περιβάλλοντα χωρίς εκτενείς τροποποιήσεις είναι πολύτιμη, μειώνοντας σημαντικά τον χρόνο και το κόστος ανάπτυξης. Στην πράξη, η **C++** βρίσκει εφαρμογή σε ποικίλους τομείς των ενσωματωμένων συστημάτων. Στην αυτοκινητοβιομηχανία, χρησιμοποιείται για την ανάπτυξη προηγμένων συστημάτων ελέγχου, συστημάτων ασφαλείας και ψυχαγωγικών συστημάτων. Στον τομέα της καταναλωτικής ηλεκτρονικής, αποτελεί τη βάση για τη λειτουργία έξυπνων συσκευών όπως κινητά τηλέφωνα, έξυπνα ρολόγια και οικιακές συσκευές. Σε βιομηχανικές εφαρμογές, η **C++** χρησιμοποιείται για τον έλεγχο μηχανημάτων και αυτοματοποιημένων γραμμών παραγωγής. Ωστόσο, ο προγραμματισμός με **C++** για ενσωματωμένα συστήματα δεν έρχεται χωρίς προκλήσεις. Οι περιορισμοί σε μνήμη και υπολογιστική ισχύ απαιτούν ιδιαίτερη προσοχή στη βελτιστοποίηση του κώδικα. Οι προγραμματιστές πρέπει να είναι σε θέση να γράφουν κώδικα που δεν είναι μόνο αποδοτικός, αλλά και αξιόπιστος και εύκολος στη συντήρηση. Αυτό απαιτεί την υιοθέτηση καλών πρακτικών προγραμματισμού, όπως η σωστή διαχείριση μνήμης, ο αποτελεσματικός χειρισμός σφαλμάτων και η χρήση κατάλληλων σχεδιαστικών προτύπων. Κοιτάζοντας προς το μέλλον, η **C++** συνεχίζει να εξελίσσεται για να ανταποκριθεί στις αυξανόμενες απαιτήσεις των ενσωματωμένων συστημάτων. Οι πρόσφατες εκδόσεις της γλώσσας έχουν εισάγει νέες δυνατότητες που την καθιστούν ακόμα πιο κατάλληλη για σύγχρονες εφαρμογές. Με την άνοδο του Διαδικτύου των Πραγμάτων (**IoT**) και την ενσωμάτωση τεχνητής νοημοσύνης σε ενσωματωμένες συσκευές, η **C++** φαίνεται να διατηρεί τη θέση της ως μία από τις πιο σημαντικές γλώσσες προγραμματισμού στον τομέα αυτό. Η εκμάθηση **C++** απαιτεί σημαντική επένδυση χρόνου και προσπάθειας λόγω της πολυπλοκότητας και του εύρους της. Ωστόσο, αυτή η επένδυση συχνά αποδίδει, καθώς οι προγραμματιστές **C++** αποκτούν βαθιά κατανόηση των θεμελιωδών αρχών προγραμματισμού και των μηχανισμών υπολογιστικών συστημάτων. Αυτή η γνώση είναι πολύτιμη ακόμη και όταν εργάζονται με άλλες γλώσσες, καθώς πολλές από τις έννοιες και τις τεχνικές που χρησιμοποιούνται στην **C++** μεταφέρονται σε διαφορετικά περιβάλλοντα προγραμματισμού. Η μοναδική της ικανότητα να συνδυάζει τον προγραμματισμό χαμηλού επιπέδου με αφαιρετικές έννοιες υψηλού επιπέδου την καθιστά αναντικατάστατη για εφαρμογές όπου η απόδοση, ο έλεγχος και η εκφραστικότητα είναι εξίσου σημαντικά. Το όραμα του **Bjarne Stroustrup** για μια γλώσσα που θα

μπορούσε να αντιμετωπίσει πολύπλοκα προβλήματα χωρίς να θυσιάζει την αποδοτικότητα έχει σίγουρα εκπληρωθεί, και η κληρονομιά της **C++** συνεχίζει να διαμορφώνει το τοπίο του προγραμματισμού υπολογιστών.[17-20]

3.2 Δυναμικές γλώσσες προγραμματισμού

Ο Δυναμικός Προγραμματισμός (**Dynamic Programming - DP**) είναι μια ισχυρή τεχνική που συνδυάζει την αποθήκευση λύσεων και την αναδρομική ανάλυση για την επίλυση πολύπλοκων προβλημάτων με αποδοτικό τρόπο. Χάρη στη μείωση της πολυπλοκότητας, αποτελεί ένα βασικό εργαλείο στον σχεδιασμό αλγορίθμων και την ανάπτυξη αποδοτικών λύσεων, καθώς είναι ιδιαίτερα αποτελεσματική για προβλήματα με επαναλαμβανόμενα υποπροβλήματα. Ο δυναμικός προγραμματισμός εφαρμόζεται σε προβλήματα με συγκεκριμένα χαρακτηριστικά δομής. Τα κύρια γνωρίσματα που καθιστούν ένα πρόβλημα κατάλληλο για επίλυση με αυτήν την προσέγγιση είναι [21,22]:

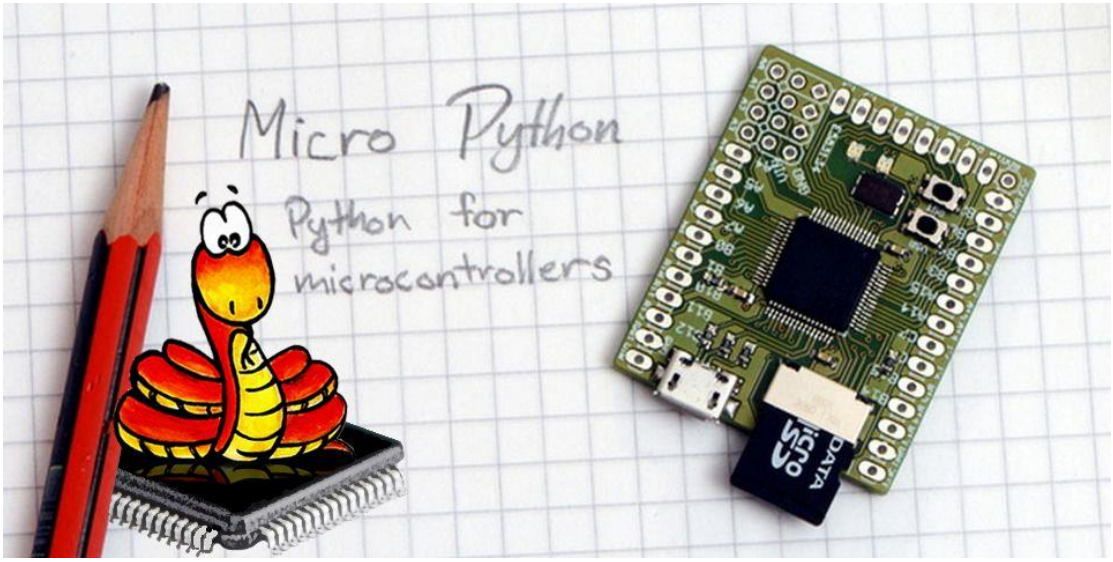
- Επικαλυπτόμενα υποπροβλήματα (**Overlapping Subproblems**)
  - Το κύριο πρόβλημα μπορεί να αναλυθεί σε μικρότερα υποπροβλήματα, τα οποία εμφανίζονται επανειλημμένα κατά τη διαδικασία επίλυσης. Αυτό δημιουργεί την ανάγκη για αποθήκευση και επαναχρησιμοποίηση των ενδιάμεσων λύσεων.
- Βέλτιστη υποδομή (**Optimal Substructure**)
  - Η βέλτιστη λύση του κύριου προβλήματος προκύπτει από τον βέλτιστο συνδυασμό των λύσεων των επιμέρους υποπροβλημάτων. Αυτή η ιδιότητα επιτρέπει τη διαίρεση του αρχικού προβλήματος σε πιο διαχειρίσιμα μέρη.

Κάποια από τα πιο γνωστά προβλήματα που επιλύονται με δυναμικό προγραμματισμό περιλαμβάνουν : Ακολουθία **Fibonacci** (Η αναδρομική προσέγγιση βασίζεται στη θεμελιώδη ιδιότητα της ακολουθίας, με την οποία κάθε όρος της ορίζεται ως το άθροισμα των δύο αμέσως προηγουμένων όρων), Εύρεση Μεγαλύτερης Κοινής Υποακολουθίας (**LCS**), Αλγόριθμοι Εύρεσης Βέλτιστης Διαδρομής (**Bellman-Ford, Floyd-Warshall**), Πρόβλημα Επεξεργασίας Απόστασης (**Edit Distance**), Βελτιστοποίηση Ακολουθίας Πολλαπλασιασμού Πινάκων (**Matrix Chain Multipplication**).[23]

ΚΕΦΑΛΑΙΟ 4 MicroPython: Θεωρητική Προσέγγιση

4.1 Τι είναι η MicroPython

Η **MicroPython** είναι μια ελαφριά και βελτιστοποιημένη έκδοση της γλώσσας προγραμματισμού **Python**, σχεδιασμένη ειδικά για την ανάπτυξη εφαρμογών σε ενσωματωμένα συστήματα με έμφαση σε μικροελεγκτές. Το όνομα της προέρχεται από τον συνδυασμό των λέξεων, **Micro** (αναφέρεται σε μικροελεγκτές, μια κατηγορία ενσωματωμένων συστημάτων) και **Python**, διατηρώντας την ευχρηστία της **Python**, αλλά προσαρμοσμένη για χρήση σε μικροσυσκευές. Λειτουργώντας ως διερμηνευτής ανοιχτού κώδικα, η **MicroPython**, επιτρέπει την εκτέλεση προγραμμάτων σε συστήματα με περιορισμένους πόρους (μικρή μνήμη, χαμηλή ισχύ), διατηρώντας παράλληλα τα βασικά χαρακτηριστικά της **Python**. Το κύριο πλεονέκτημά της είναι η δυνατότητα ανάπτυξης κώδικα υψηλού επιπέδου για τον έλεγχο υλικών συστημάτων, προσφέροντας μια ευέλικτη εναλλακτική λύση έναντι των παραδοσιακών γλωσσών όπως **C** ή **Assembly**. Αυτές οι ιδιότητες την καθιστούν ιδανική για εκπαιδευτικούς σκοπούς, αποτελεί ένα ιδανικό περιβάλλον εκμάθησης, καθώς επιτρέπει σε αρχάριους να κατανοήσουν τόσο τις θεμελιώδεις αρχές του προγραμματισμού (βρόχοι, συναρτήσεις) όσο και τις βασικές έννοιες του προγραμματισμού ενσωματωμένων συστημάτων (ανάγνωση αισθητήρων).[24,25]



Εικόνα 3: MicroPython Logo (https://github.com/micropython/micropython)

4.2 Βασικές διαφορές με την Python

Σκοπός και Περιβάλλον Εκτέλεσης

- **Python**: Γλώσσα υψηλού προγραμματισμού που σχεδιάστηκε για γενικούς υπολογιστικούς σκοπούς, εκτελείται σε πλήρης λειτουργικά συστήματα και απαιτεί σημαντικούς πόρους (**CPU**, Μνήμη)

- **MicroPython**: Ελαφριά και βελτιστοποιημένη έκδοση της **python**, σχεδιασμένη ειδικά για ενσωματωμένα συστήματα και μικροελεγκτές με ελάχιστες απαιτήσεις σε πόρους (**Raspberry Pi Pico, ESP32**)

Διαχείριση μνήμης
- **Python**: Χρησιμοποιεί αυτόματη διαχείριση μνήμης (**garbage collector**)
- **MicroPython**: Πιο αυστηρή και απαιτεί μεγαλύτερη προσοχή στη δέσμευση και απελευθέρωση πόρων

Ενσωμάτωση με υλικό
- **Python**: Δεν είναι κατάλληλη για άμεση επικοινωνία με χαμηλού επιπέδου υλικό, εκτός αν χρησιμοποιηθούν εξωτερικές βιβλιοθήκες (**RPi.GPIO για Raspberry Pi**)
- **MicroPython**: Επιτρέπει άμεση επικοινωνία με υλικό (αισθητήρες, οθόνες) μέσω απλών βιβλιοθηκών

Σύνταξη
- Κοινά Στοιχεία: Οι δύο γλώσσες χρησιμοποιούν εσοχές για δομές ελέγχου και έχουν παρόμοια βασική σύνταξη
- **MicroPython**: Περιορισμένη υποστήριξη για ορισμένες προχωρημένες δυνατότητες

Βιβλιοθήκες
- **Python**: Εκτεταμένη συλλογή βιβλιοθηκών (**pandas, tensorflow**)
- **MicroPython**: Περιορισμένες βιβλιοθήκες, ωστόσο επιτρέπει τη δημιουργία ‘παγωμένων’ (**frozen**) modules για εξοικονόμηση μνήμης

Απαιτήσεις πόρων
- **Python**: Απαιτεί σημαντικούς πόρους μνήμης
- **MicroPython**: Καταναλώνει ελάχιστη μνήμη

Χρήση
- **Python**: Επιστημονικός προγραμματισμός, ανάλυση δεδομένων
- **MicroPython**: Ενσωματωμένα συστήματα (**IoT**, ρομποτική)

Διερμηνευτής
- **Python**: Πλήρως λειτουργικός διερμηνευτής
- **MicroPython**: Διερμηνευτής ελάχιστου μεγέθους για συστήματα με περιορισμένους πόρους

Υποστήριξη Συστήματος Αρχείων
- **Python**: Πλήρης πρόσβαση σε αρχεία (τοπικός χώρος αποθήκευσης και δίκτυο)
- **MicroPython**: Περιορισμένη υποστήριξη αρχείων (αποθήκευση **flash**)

Ταχύτητα Εκτέλεσης
- **Python**: Παρέχει ταχύτερη εκτέλεση σε γενικής χρήσης υλικό

- **MicroPython**: Μειωμένη ταχύτητα λόγω των περιορισμένων πόρων που έχουν οι μικροελεγκτές

Χειρισμός Σφαλμάτων
- **Python**: Ολοκληρωμένη διαχείριση σφαλμάτων και εργαλεία διόρθωσης λαθών (**debug**)
- **MicroPython**: Περιορισμένα εργαλεία αποσφαλμάτωσης και λιγότερους τύπους εξαιρέσεων

Περιβάλλον Ανάπτυξης
- **Python**: Προηγμένα εργαλεία ανάπτυξης, όπως ολοκληρωμένα περιβάλλοντα (**IDEs**), αποσφαλματωτές (**debuggers**) και αναλυτές απόδοσης (**profilers**)
- **MicroPython**: Χρησιμοποιεί συνήθως απλούστερα περιβάλλοντα ανάπτυξης (**Thonny, uPyCraft**), ενώ προσφέρει άμεση αλληλεπίδραση μέσω του **REPL** (**Read-Eval-Print Loop**)

Η **MicroPython** και η **Python**, παρόλο που μοιράζονται κοινή σύνταξη και φιλοσοφία εξυπηρετούν εντελώς διαφορετικούς σκοπούς στον χώρο του προγραμματισμού. Η **MicroPython** αποτελεί μια αποδοτικά σχεδιασμένη υλοποίηση της **Python**, σχεδιασμένη για ενσωματωμένα συστήματα και μικροελεγκτές. Με ελάχιστες απαιτήσεις πόρων (**16KB** μνήμης **RAM** και **256KB Flash**), προσφέρει μια ευέλικτη πλατφόρμα για ανάπτυξη έξυπνων συσκευών, εφαρμογών **IoT** και ρομποτικών συστημάτων. Η ικανότητα της να διαχειρίζεται άμεσα το υλικό μέσω απλών εντολών **Python** την καθιστά ιδανική για γρήγορα πρωτότυπα και εκπαιδευτικές εφαρμογές. Από την άλλη πλευρά, η **Python** κυριαρχεί σε τομείς που απαιτούν υπολογιστική ισχύ και ευελιξία. Στην επιστημονική έρευνα, την ανάλυση δεδομένων, την τεχνητή νοημοσύνη και την ανάπτυξη εφαρμογών, η **Python** αποτελεί την πρώτη επιλογή χάρη στο τεράστιο οικοσύστημα βιβλιοθηκών της. Παρόλο που η **Python** καταναλώνει περισσότερους πόρους σε σύγκριση με τη **MicroPython**, αυτό δεν αποτελεί πρόβλημα στους τομείς εφαρμογής της, όπου η διαθεσιμότητα πόρων είναι συνήθως μεγαλύτερη.[26-28]

4.3 Υποστηριζόμενες πλατφόρμες (ESP32, Raspberry Pi Pico, STM32 κ.ά.)

Το σύγχρονο πεδίο των ενσωματωμένων συστημάτων χαρακτηρίζεται από μια ευρεία ποικιλία υλικών πλατφορμών, καθεμία με ιδιαίτερα χαρακτηριστικά και δυνατότητες. Σε αυτό το πλαίσιο, η **MicroPython** έχει αναδειχθεί ως μια από τις πλέον αποδοτικές λύσεις ανάπτυξης, συνδυάζοντας την ευκολία της γλώσσας **Python** με την αποδοτικότητα που απαιτούν τα ενσωματωμένα συστήματα. Υποστηρίζοντας μια μεγάλη γκάμα μικροελεγκτών από διάφορους κατασκευαστές, η **MicroPython** προσφέρει ευελιξία ανάλογα με τις ανάγκες της εκάστοτε εφαρμογής. Μεταξύ των πιο δημοφιλών είναι οι [29]:

- **ESP8266/ ESP32**

  Ο **ESP8266** αποτελεί έναν οικονομικό μικροελεγκτή με ενσωματωμένες δυνατότητες **WiFi** και δικτύωσης **TCP/IP**, ο οποίος αναπτύχθηκε από την κινέζικη εταιρεία

Espressif Systems. Έχοντας κερδίσει ευρεία αποδοχή από την εμφάνιση του το 2014, έδωσε σημαντική ώθηση στην ανάπτυξη εφαρμογών **IoT** (**Internet of Things**) χαμηλού κόστους. Η ανάδειξη της **MicroPython** ως εργαλείου ανάπτυξης εφαρμογών για αυτόν τον μικροελεγκτή απλοποίησε περαιτέρω τη δημιουργία έργων, μειώνοντας την πολυπλοκότητα προγραμματισμού σε συστήματα με περιορισμένους πόρους. Με στόχο την κάλυψη των εξελισσόμενων απαιτήσεων του **IoT**, η **Espressif Systems** εισήγαγε τον νεότερο μικροελεγκτή **ESP32**, ο οποίος συνδυάζει βελτιωμένες λειτουργίες με αυξημένη υπολογιστική ισχύ. Πέρα από το **WiFi**, ο **ESP32** υποστηρίζει και την δυνατότητα **Bluetooth**, καθώς η διπύρηνη αρχιτεκτονική του βελτιώνει σημαντικά την απόδοση σε εφαρμογές πραγματικού χρόνου. Επιπλέον, διαθέτει ειδικό υλικό για κρυπτογραφημένες λειτουργίες και ασφαλή δημιουργία τυχαίων αριθμών, χαρακτηριστικά που τον καθιστούν ιδανικό για εφαρμογές **IoT**. Παρόλο που η υλοποίηση της **MicroPython** για τον **ESP32** βρίσκεται ακόμη σε στάδιο ανάπτυξης, η αρχιτεκτονική του διατηρεί υψηλή συμβατότητα με τον **ESP8266**. Αυτό σημαίνει ότι ο κώδικας που αναπτύσσεται για τον **ESP8266** μπορεί να μεταφερθεί στον **ESP32** χωρίς σημαντικές τροποποιήσεις, επιτρέποντας σταδιακή μετάβαση μεταξύ πλατφορμών.[30,31]

- **Raspberry Pi Pico**

  Το **Raspberry Pi Pico** αποτελεί μια οικονομικά προσιτή πλακέτα ενσωματωμένης επεξεργασίας, η οποία ενδείκνυται για χρήση σε εκπαιδευτικά περιβάλλοντα, ιδιαίτερα σε πανεπιστημιακά προγράμματα μηχανικών και τεχνολογικών επιστημόνων. Η πλακέτα αυτή βασίζεται στον μικροελεγκτή **RP2040** και υλοποιείται με αρχιτεκτονική **system- on-a-chip** (**SoC**), παρέχοντας τις βασικές λειτουργιές ενός πλήρους υπολογιστικού συστήματος σε ενιαίο ολοκληρωμένο κύκλωμα. Το **Raspberry Pi Pico** διαθέτει ένα ευρύ φάσμα τεχνολογικών χαρακτηριστικών που το καθιστούν ιδανικό για εκπαιδευτικές και ερευνητικές εφαρμογές. Συγκεκριμένα, η πλακέτα προσφέρει πλούσιο σύνολο ψηφιακών περιφερειακών, το οποίο περιλαμβάνει 2 διεπαφές **SPI**, 2 **I2C**, 2 **UART** και 16 κανάλια , επιτρέποντας την εύκολη σύνδεση με ποικίλες εξωτερικές συσκευές. Επιπλέον, διαθέτει 23 προγραμματιζόμενους ακροδέκτες εισόδου/εξόδου (**GPIO**) και 3 αναλογικές εισόδους (**ADC**) με δυνατότητα μετατροπής σήματος. Η παρουσία ενσωματωμένων στοιχείων, όπως ενός **LED** και αισθητήρα θερμοκρασίας, προσφέρει άμεσες δυνατότητες δοκιμής και αξιολόγησης βασικών λειτουργιών του συστήματος. Από πλευράς ανάπτυξης λογισμικού, η πλακέτα παρέχει υποστήριξη για προγραμματισμό σε **C** και **MicroPython**, καλύπτοντας τόσο απαιτητικές όσο και εκπαιδευτικές εφαρμογές. Προκειμένου να διευκολυνθεί η διαδικασία ανάπτυξης, το ολοκληρωμένο περιβάλλον **Thonny** προσφέρει σημαντική βοήθεια για τη συγγραφή, αποσφαλμάτωση και εκτέλεση κώδικα, με τη δυνατότητα άμεσης μεταφόρτωσης στον μικροελεγκτή. [32-34]

- **STM32**

  Βασισμένοι στην ισχυρή αρχιτεκτονική **ARM Cortex-M**, οι μικροελεγκτές **STM32** προσφέρουν μια ευρεία γκάμα επιλογών στην κατηγορία των **32-bit** συστημάτων ενσωματωμένου ελέγχου. Οι μικροελεγκτές αυτοί διακρίνονται για τον συνδυασμό σημαντικών χαρακτηριστικών, γεγονός που τους καθιστά κατάλληλους σε πολλαπλά πεδία χρήσης. Συγκεκριμένα, προσφέρουν υψηλή υπολογιστική ισχύ, δυνατότητες επεξεργασίας σε πραγματικό χρόνο και ενσωματωμένη υποστήριξη ψηφιακής επεξεργασίας σήματος (**DSP**). Παράλληλα, είναι σχεδιασμένοι να καταναλώνουν ελάχιστη ενέργεια και να λειτουργούν σε χαμηλές τάσεις, διατηρώντας τη δυνατότητα ταχείας ανάπτυξης και υποστήριξης σύνθετων λειτουργιών. Οι τεχνολόγοι μπορούν να υλοποιήσουν τόσο ανεξάρτητα έργα όσο και πλήρως ανεπτυγμένες πλατφόρμες, επωφελούμενοι από τη δυνατότητα εύκολης επαναχρησιμοποίησης και μεταφοράς του κώδικα μεταξύ διαφορετικών σειρών **STM32**. Αυτό το χαρακτηριστικό συμβάλει αποφασιστικά στη μείωση του χρόνου που απαιτείται για την ένταξη των προϊόντων στην αγορά. Για την έναρξη της ανάπτυξης εφαρμογών με **STM32**, η **STMicroelectronics** προσφέρει μια ολοκληρωμένη σειρά πλακετών αξιολόγησης.[35,36]

Άλλες σημαντικές πλατφόρμες είναι:

- **RP2040**
- **Teensy 3.X και Teensy 4.X**
- **BBC micro: bit**
- **ARM Cortex-M**
- **Zephyr**
- **Arduino** (συγκεκριμένα μοντέλα)

4.4 Περιβάλλον ανάπτυξης και εργαλεία

Η δημιουργία εφαρμογών σε **MicroPython** είναι εφικτή ακόμη και μέσω βασικών επεξεργαστών κειμένου. Ωστόσο, η αξιοποίηση ενός Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης (**IDE**) διευκολύνει σημαντικά τη διαδικασία.[37]

- Ο χρωματικός διαχωρισμός της σύνταξης (**Syntax Highlighting**) συμβάλει στη βελτίωση της αναγνωσιμότητας του κώδικα, επισημαίνοντας με διαφορετικά χρώματα τις λέξεις-κλειδιά τις μεταβλητές, τις συμβολοσειρές και τα σχόλια. Η δυνατότητα αυτή διευκολύνει τον προγραμματιστή στον άμεσο εντοπισμό σφαλμάτων και ενισχύει τη συνολική κατανόηση της δομής του προγράμματος.
- Η λειτουργία αυτόματης συμπλήρωσης κώδικα (**Intellisense**) συμβάλλει στη μείωση των λαθών και επιταχύνει την ανάπτυξη, προτείνοντας εντολές ή μεταβλητές καθώς ο χρήστης πληκτρολογεί. Με αυτό τον τρόπο, διευκολύνεται η εργασία με βιβλιοθήκες ή συναρτήσεις που δεν είναι άμεσα οικείες στον χρήστη.

- Τα εργαλεία αποσφαλμάτωσης (**Debugging Tools**) παρέχουν τη δυνατότητα εκτέλεσης του προγράμματος βήμα προς βήμα, επιτρέποντας την επιθεώρηση της κατάστασης των μεταβλητών και τον εντοπισμό σφαλμάτων κατά την εκτέλεση. Η διαδικασία αυτή καθίσταται απαραίτητη για την επίλυση προβλημάτων σε σύνθετες εφαρμογές.
- Η λειτουργία διαχείρισης αρχείων προσφέρει μια αποτελεσματική οργάνωση και εποπτεία των αρχείων ενός **project**, διευκολύνοντας τον εντοπισμό, την επεξεργασία και την δομή του κώδικα.
- Μέσω της ενσωματωμένης σειριακής διεπαφής (**Serial Monitor Integration**), επιτρέπει την άμεση επικοινωνία με τον μικροελεγκτή, προσφέροντας χρήσιμες πληροφορίες για τον έλεγχο και την αποσφαλμάτωση του συστήματος.
- Παρέχει εύχρηστο μηχανισμό για την εγκατάσταση του **firmware**, επιτρέποντας άμεση μεταφόρτωση του προγράμματος στη συσκευή.

Τα πιο διάσημα **IDEs**:

- Το **Thonny** αποτελεί ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης (**IDE**), σχεδιασμένο για να διευκολύνει τη συγγραφή και αποσφαλμάτωση κώδικα σε **MicroPython**. Η απλότητα στη ρύθμιση και λειτουργία του το καθιστά ιδανικό για αρχάριους, ενώ ταυτόχρονα υποστηρίζει πλήρως τη σύνδεση με δημοφιλείς πλατφόρμες, όπως οι **ESP32** και **Raspberry Pi Pico**. Επιπλέον, παρέχει τη δυνατότητα σύνταξης και αποστολής κώδικα απευθείας στη συσκευή, καθιστώντας τη διαδικασία ανάπτυξης πιο άμεση και αποδοτική.[38]
- Το **Visual Studio Code**, σε συνδυασμό με την επέκταση **Pymakr**, προσφέρει ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τον σχεδιασμό και την υλοποίηση εφαρμογών **MicroPython** σε πλακέτες **ESP32** και **ESP8266**. Η λύση αυτή απευθύνεται κυρίως σε χρήστες με προγραμματιστική εμπειρία, καθώς παρέχει προηγμένες δυνατότητες και μεγαλύτερη ευελιξία σε σύγκριση με απλούστερα εργαλεία. Η επέκταση **Pymakr** επιτρέπει την απευθείας επικοινωνία μεταξύ του **Visual Studio Code** και των συσκευών **MicroPython**, μέσω της διεπαφής **REPL** (**Read-Eval-Print-Loop**), δίνοντας τη δυνατότητα εκτέλεσης μεμονωμένων αρχείων, συγχρονισμού ολόκληρων **projects** ή απευθείας πληκτρολόγησης και εκτέλεσης εντολών μέσω του ενσωματωμένου τερματικού του **VS Code**.[39]
- Το **uPyCraft** είναι ένα ελαφρύ ολοκληρωμένο περιβάλλον ανάπτυξης για τη δημιουργία εφαρμογών **MicroPython** σε μικροελεγκτές **ESP32** και **ESP8266**. Αναπτύχθηκε από την **DFRobot** και ξεχωρίζει για την απλότητα του, καθιστώντας το ιδανικό για αρχάριους. Το **uPyCraft** προσφέρει βασικές λειτουργίες, όπως διαχείριση αρχείων, ενσωματωμένο τερματικό **REPL**, σύνδεση μέσω **USB** και βασικό **debugging**. Σε σύγκριση με **IDEs** όπως το **Thonny** ή το **VS Code**, υπερτερεί σε ευκολία χρήσης, αλλά υπολείπεται σε προηγμένες δυνατότητες αποσφαλμάτωσης.[40]

Η **MicroPython** περιλαμβάνει ένα διαδραστικό περιβάλλον γραμμής εντολών (**REPL**), το οποίο εκτελείται σε μικροελεγκτές όπως οι **ESP32**, **Raspberry Pi Pico** και **BBC Micro:bit**. Η αλληλεπίδραση με το περιβάλλον αυτό πραγματοποιείται μέσω εργαλείων σειριακής επικοινωνίας, τα οποία λειτουργούν ως διασύνδεση μεταξύ του υπολογιστή και της μικροελεγκτική πλακέτας.[41-44]

- Το **rshell** αποτελεί ένα εργαλείο γραμμής εντολών σχεδιασμένο για την απομακρυσμένη διαχείριση συσκευών που τρέχουν **MicroPython**. Επικοινωνεί με την πλακέτα μέσω της λειτουργίας **raw-REPL**, προσφέροντας πρόσβαση στο περιβάλλον εντολών αλλά και δυνατότητες διαχείρισης αρχείων. Μπορεί να λειτουργήσει ακόμα και όταν η λειτουργία **USB Mass Storage** είναι απενεργοποιημένη.
- Το **pyboard.py** είναι ένα αυτόνομο εργαλείο γραμμένο σε **Python**, το οποίο εκτελείται στον υπολογιστή και επιτρέπει την άμεση εκτέλεση εντολών και σεναρίων (**scripts**) στη συσκευή, χωρίς να απαιτείται προηγούμενη αντιγραφή αρχείων. Παρέχει επίσης πλήρη πρόσβαση στο σύστημα αρχείων της συσκευής, ακόμη και όταν δεν υποστηρίζεται η λειτουργία **USB Mass Storage**. Παρά την ονομασία του, είναι συμβατό με όλες τις πλατφόρμες **MicroPython**.
- Το **ampy** (**Adafruit MicroPython Tool**) αποτελεί ένα ελαφρύ εργαλείο γραμμής εντολών, σχεδιασμένο για βασική διαχείριση και έλεγχο συσκευών που τρέχουν **MicroPython**. Παρέχει τρεις κύριες λειτουργίες: (α) αποστολή αρχείων από τον υπολογιστή προς την πλακέτα, (β) λήψη αρχείων από την πλακέτα στον υπολογιστή, (γ) εκτέλεση **Python scripts** απευθείας στην πλακέτα.
- Το **PuTTY** είναι ένα δωρεάν και ανοιχτού κώδικα εργαλείο τερματικού, σχεδιασμένο για απομακρυσμένη πρόσβαση και επικοινωνία με συστήματα μέσω πρωτοκόλλων όπως **SSH**, **Telnet** και **login**. Χρησιμοποιείται ευρέως για τη διαχείριση απομακρυσμένων υπολογιστών και συσκευών, ενώ ξεχωρίζει για την απλότητα εγκατάστασης, τη χαμηλή κατανάλωση πόρων και τις ασφαλείς μεθόδους κρυπτογράφησης. Η ευελιξία του στη ρύθμιση παραμέτρων και η υποστήριξη του από την κοινότητα το καθιστούν ιδιαίτερα χρήσιμο για προγραμματιστές, διαχειριστές συστημάτων και χρήστες **MicroPython**.

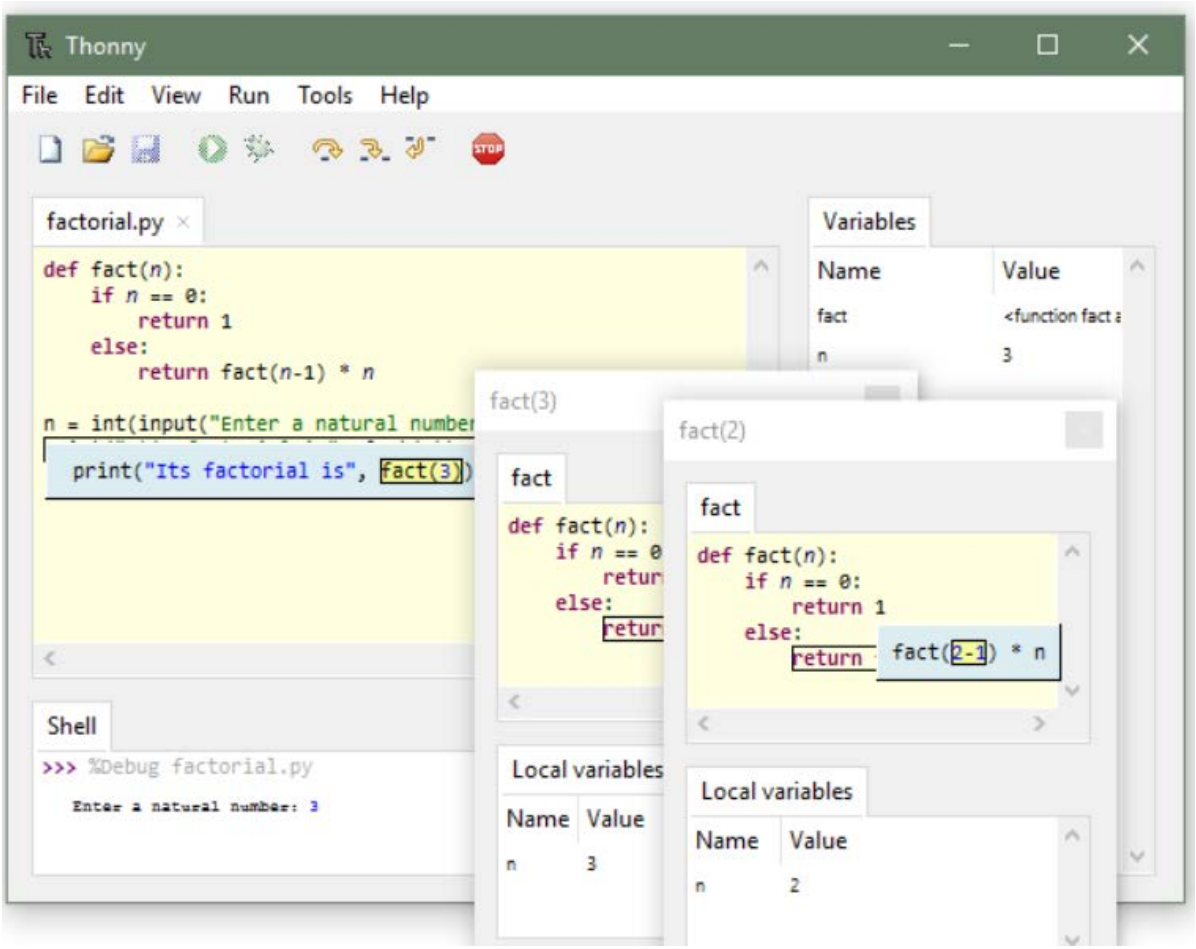
ΚΕΦΑΛΑΙΟ 5 Προγραμματισμός με MicroPython

5.1 Εγκατάσταση και ρύθμιση περιβάλλοντος ανάπτυξης

Για την εύκολη και αποτελεσματική εγκατάσταση της **MicroPython**, συνιστάται η χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης (**IDE**) **Thonny**, το οποίο διαθέτει φιλικό προς το χρήστη περιβάλλον και είναι ιδανικό για αρχάριους αλλά και πιο προχωρημένους χρήστες.[45]

1. Επισκεφθείτε τον επίσημο ιστότοπο του **Thonny** στη διεύθυνση https://thonny.org/.
2. Κατεβάστε την κατάλληλη έκδοση του προγράμματος για το λειτουργικό σας σύστημα (**Windows, macOS ,Linux**).
3. Ακολουθήστε τις τυπικές διαδικασίες του συστήματος σας.

Η χρήση του **Thonny** εξασφαλίζει μια οργανωμένη και αποδοτική εμπειρία ανάπτυξης, ιδιαίτερα χρήσιμη σε έργα που απαιτούν προγραμματισμό ενσωματωμένων συστημάτων, όπως αυτά με χρήση **MicroPython** και μικροελεγκτές (π.χ. **Raspberry Pi Pico**).



Εικόνα 4: Περιβάλλον Thonny, https://thonny.org/

Στη συνέχεια:

1. Εκκινήστε το **Thonny**.
2. Επιβεβαιώστε ότι λειτουργεί στην κανονική λειτουργία (**standard mode**). Αν εμφανιστεί μωβ υπερσύνδεσμος με σχετική ειδοποίηση, κάντε κλικ πάνω του.
3. Ακολουθήστε την οθόνη που εμφανίζεται, κλείστε το **Thonny** και ξανανοίξτε το για να τεθούν σε ισχύ οι αλλαγές.
4. Πλοηγηθείτε στο μενού εγκατάστασης επιλέγοντας '**Install MicroPython**..' από την κάτω δεξιά γωνία του **Thonny**.
5. Διαλέξτε την έκδοση που ταιριάζει με τη συσκευή σας. Για παράδειγμα, στην περίπτωση του **Raspberry Pi Pico**, θα πρέπει να επιλέξετε την επιλογή **'Raspberry Pi – Pico / Pico H'**.
6. Εκκινήστε τη διαδικασία με το κουμπί '**Install'**, ώστε να ξεκινήσει η μεταφόρτωση της **MicroPython** στον μικροελεγκτή σας.

5.2 Βασικές εντολές και δομές προγραμματισμού

- Βασικές Εντολές:

  Εκτύπωση: **print("Hello, world!")**

  Σχόλιο: **#Hello World**

  Μαθηματικές Πράξεις: **a** = 5 + 3, **b** = 10 / 2

- Δομές Ελέγχου:

  Συνθήκες (**if, elif, else**):

```
if x ==10:
    print('x is 10')
elif x <=10:
    print('x is less than 10')
else:
    print('x is not 10')
```

- Βρόχοι (**for, while**)

```
# For loop
for i in range(5):   # 0 έως 4
    print('i is:' ,i)

# While loop
count = 0
while count < 5:
    print(count)
    count +=1
```

5.3 Χειρισμός εισόδου/εξόδου (GPIO, LED, κουμπιά)

- Οι ακίδες **GPIO** (**General Purpose Input/Output**) επιτρέπουν τη σύνδεση και έλεγχο εξωτερικών συσκευών. Στη **MicroPython**, η διαχείρισή τους γίνεται μέσω της κλάσης **Pin** από την ενότητα **machine**. Κάθε αντικείμενο **Pin** υλοποιεί ένα σύνολο μεθόδων που επιτρέπουν τον ορισμό της λειτουργικής κατάστασης κάθε ακίδας (εισόδου, εξόδου ή άλλων ειδικών λειτουργιών), καθώς και την ανάγνωση και τροποποίηση της ψηφιακής της κατάστασης. Σημαντική είναι η επαλήθευση διαθεσιμότητας ακίδων, καθώς ορισμένες μπορεί να χρησιμοποιούνται για ειδικές λειτουργίες όπως για παράδειγμα στην πλατφόρμα **ESP32**, οι ακίδες 0, 2, 4, 5, 12-16 είναι συνήθως διαθέσιμες για γενική χρήση, αν και κάποιες από αυτές (όπως η GPIO0 ή GPIO16) ενδέχεται να επηρεάζουν τη διαδικασία εκκίνησης και απαιτούν προσεκτική χρήση. Η σωστή διαχείριση των **GPIO** μέσω της κλάσης Pin είναι απαραίτητη για την ανάπτυξη αξιόπιστων εφαρμογών.[46,47]

- Η διαδικασία αναβοσβήσματος ενός **LED** μπορεί να πραγματοποιηθεί απευθείας μέσω της διαδραστικής διεπαφής **REPL** που παρέχει η **MicroPython**, χωρίς την ανάγκη δημιουργίας και μεταφόρτωσης αρχείων κώδικα. Το **REPL** λειτουργεί ως περιβάλλον απευθείας εκτέλεσης εντολών **Python**, επιτρέποντας στον χρήστη να γράφει και να δοκιμάζει κώδικα σε πραγματικό χρόνο.[48]

```
import machine # Εισάγει τη βιβλιοθήκη machine, η οποία έχει πρόσβαση στις λειτουργίες του υλικού (όπως GPIO)
import time # Εισάγει τη βιβλιοθήκη time, η οποία παρέχει συναρτήσεις για χρονικές καθυστερήσεις


led = machine.Pin(2, Pin.OUT) # Ορίζει τον ακροδέκτη GPIO2 ως έξοδο (Pin.OUT) και τον αποθηκεύει στη μεταβλητή led
while True: #Δημιουργεί έναν άπειρο βρόχο, οπότε ο κώδικας μέσα σε αυτόν θα εκτελείται συνεχώς
    led.value(1) #Ενεργοποιεί την LED
    time.sleep(1) #Παύση 1 δευτερόλεπτο (η LED παραμένει αναμμένη)
    led.value(0) #Απενεργοποιεί την LED
    time.sleep(1) #Παύση 1 δευτερόλεπτο (η LED παραμένει σβηστή)
```

[49]

- Σε περίπτωση που μια ψηφιακή ακίδα εισόδου/εξόδου (**I/O**) παραμένει αποσυνδεδεμένη από οποιαδήποτε εξωτερική σύνδεση, η συμπεριφορά της καθίσταται απρόβλεπτη. Αυτό οφείλεται στο γεγονός ότι η ακίδα βρίσκεται σε 'πλωτή' κατάσταση (**floating**), δηλαδή χωρίς καθορισμένο λογικό επίπεδο, γεγονός που μπορεί να οδηγήσει σε τυχαία ανάγνωση τιμών **HIGH** ή **LOW**. Για την αποφυγή αυτής της αστάθειας, είναι απαραίτητη η χρήση αντιστάσεων **pull-up** ή **pull-down**, οι οποίες καθορίζουν σταθερά το λογικό επίπεδο της ακίδας όταν αυτή δεν εφαρμόζεται εξωτερικό σήμα. Στην αδρανή κατάσταση, όταν ο διακόπτης παραμένει ανοιχτός, η ψηφιακή είσοδος παραμένει συνδεδεμένη με την τάση τροφοδοσίας(π.χ. **5V**), καταγράφοντας σταθερά λογική τιμή **HIGH**. Όταν ο διακόπτης πατηθεί, κλείσει το κύκλωμα και η είσοδος συνδέεται με τη γείωση(**GND**), με αποτέλεσμα να καταγράφεται η λογική τιμή **LOW**. Με αυτό τον τρόπο, διασφαλίζεται η αξιοπιστία και σταθερή ανάγνωση της κατάστασης του διακόπτη, αποφεύγοντας παρεμβολές ή τυχαίες ενδείξεις.[50]

```
from machine import Pin #Εισάγει την κλάση Pin από το module machine, που χρησιμοποιείται για διαχείριση των GPIO


led = Pin(5, Pin.OUT) #Ορίζει τον ακροδέκτη GPIO5 ως έξοδο (Pin.OUT) και τον αποθηκεύει στη μεταβλητή led
btn = Pin(0, Pin.IN) #Ορίζει τον ακροδέκτη GPIO0 ως είσοδο (Pin.IN) και τον αποθηκεύει στη μεταβλητή btn

while True: #Δημιουργεί έναν άπειρο βρόχο, ώστε ο κώδικας να εκτελείται συνεχώς
    if btn() == 0: #Ελέγχει αν το κουμπί είναι πατημένο(το κουμπί συνδέεται με GND, η είσοδος διαβάζει 0 (LOW))
            led.on() #Αν το κουμπί είναι πατημένο, ανάβει την LED (θέτει την έξοδο GPIO5 σε HIGH)
    else: #Αλλιώς αν το κουμπί δεν είναι πατημένο
            led.off() #Σβήνει την LED (θέτει την έξοδο GPIO5 σε LOW)
```

[53]

5.4 Επικοινωνία με αισθητήρες (I2C, SPI, UART)

- Η τεχνολογία **I2C** (**Inter-Integrated Circuit**) αποτελεί ένα από τα πιο διαδεδομένα πρωτόκολλα σειριακής επικοινωνίας σε ενσωματωμένα συστήματα, καθώς προσφέρει τη δυνατότητα σύνδεσης μικροελεγκτών με διάφορες περιφερειακές συσκευές μέσω μόνο δύο γραμμών, καθιστώντας την ιδανική για εφαρμογές με περιορισμένο αριθμό ακίδων εισόδου/εξόδου. Η επικοινωνία μέσω **I2C** βασίζεται σε δύο γραμμές: τη γραμμή δεδομένων (**SDA**) και τη γραμμή ρολογιού (**SCL**), οι

οποίες είναι κοινές για όλες τις συσκευές στο δίαυλο και απαιτούν εξωτερικές αντιστάσεις **pull-up** για σωστή λειτουργία. Το πρωτόκολλο λειτουργεί με λογική **Master–Slave**, όπου ο **Master** ελέγχει τον συγχρονισμό και ξεκινά την επικοινωνία ενώ οι **Slaves** ανταποκρίνονται στις εντολές του. Κάθε **Slave** διαθέτει μοναδική διεύθυνση, επιτρέποντας στο σύστημα να απευθύνεται επιλεκτικά σε κάθε συσκευή. Η ανταλλαγή δεδομένων πραγματοποιείται σε μορφή πλαισίων 8 **bit**, ακολουθούμενα από ένα **bit** επιβεβαίωσης (**ACK**), και ξεκινά πάντα με σήμα έναρξης (**START**) και ολοκληρώνεται με σήμα τερματισμού (**STOP**). Ο **Master** στέλνει αρχικά τη διεύθυνση του **Slave** μαζί με ένα **bit** που δηλώνει αν πρόκειται να γίνει ανάγνωση ή εγγραφή, και στη συνέχεια ακολουθεί η μεταφορά των δεδομένων. Η επικοινωνία μπορεί να υποστηρίζει πολλαπλούς **Slaves** και ακόμη και πολλαπλούς **Masters**, με την προϋπόθεση ότι εφαρμόζεται κατάλληλος μηχανισμός διαιτησίας για αποφυγή συγκρούσεων. Ένα σημαντικό πλεονέκτημα του **I2C** είναι η απλότητά του και η δυνατότητα διασύνδεσης πολλών συσκευών με ελάχιστη καλωδίωση, γεγονός που μειώνει το κόστος και την πολυπλοκότητα σχεδίασης. Ωστόσο, παρουσιάζει και κάποιους περιορισμούς, όπως η σχετικά χαμηλή ταχύτητα μεταφοράς δεδομένων σε σύγκριση με άλλα πρωτόκολλα όπως το **SPI**, καθώς και η περιορισμένη απόσταση επικοινωνίας λόγω της φύσης της γραμμής και της ανάγκης για αντιστάσεις **pull-up**. Το πρωτόκολλο **I2C** αναπτύχθηκε αρχικά το 1982 από την εταιρεία **Philips** για χρήση εντός τηλεοράσεων, αλλά η λειτουργικότητα και η αποδοτικότητα της οδήγησαν σε ταχεία υιοθέτηση από ένα ευρύ φάσμα κατασκευαστών και εφαρμογών. Η αρχική έκδοση υποστήριζε μόνο ταχύτητες έως **100kHz** και διευθυνσιοδότηση **7-bit**, κάτι που περιόριζε τον αριθμό των διαθέσιμων διευθύνσεων για συσκευές στο δίαυλο σε 112, αφού κάποιες διευθύνσεις είναι δεσμευμένες και δεν χρησιμοποιούνται ποτέ για κανονική επικοινωνία. Το 1992 δημοσιεύτηκε η πρώτη δημόσια προδιαγραφή, η οποία εισήγαγε τη λειτουργία **fast-mode** στα **400kHz** και υποστήριξη για **10-bit** διευθύνσεις, επεκτείνοντας σημαντικά τη δυνατότητα διασύνδεσης περισσότερων συσκευών. Πολλοί μικροελεγκτές, όπως ο **ATmega328** που χρησιμοποιείται σε πολλές πλακέτες συμβατές με **Arduino**, υποστηρίζουν το **I2C** μέχρι αυτό το σημείο. Ωστόσο, στη συνέχεια προστέθηκαν επιπλέον λειτουργίες που διευρύνουν περαιτέρω τις δυνατότητες του πρωτοκόλλου. Συγκεκριμένα, η λειτουργία **fast-mode plus** επιτρέπει ταχύτητες έως **1MHz**, η λειτουργία **high-speed mode** φτάνει τα **3,4MHz**, ενώ η **ultra-fast mode** επιτρέπει θεωρητικά ταχύτητες έως και **5MHz**. Το 1995 η **Intel** εισήγαγε μια παραλλαγή που ονομάζεται **System Management Bus** (**SMBus**), η οποία βασίζεται στο **I2C** αλλά χαρακτηρίζεται από αυστηρότερες προδιαγραφές, με σκοπό τη διασφάλιση προβλέψιμης επικοινωνίας μεταξύ υποστηρικτικών ολοκληρωμένων κυκλωμάτων σε μητρικές πλακέτες υπολογιστών. Στην πράξη, το **I2C** συναντάται σε πληθώρα εφαρμογών, από αισθητήρες θερμοκρασίας, υγρασίας και επιτάχυνσης, έως οθόνες, **EEPROMs** και μικροελεγκτές. Η ευρεία αποδοχή του πρωτοκόλλου ενισχύεται από την ύπαρξη εύχρηστων βιβλιοθηκών, όπως η **Wire**

στο **Arduino**, που απλοποιούν σημαντικά την υλοποίηση της επικοινωνίας. Παρότι παρουσιάζει περιορισμούς όπως σχετικά μικρές ταχύτητες και περιορισμένες αποστάσεις επικοινωνίας, παραμένει εξαιρετικά χρήσιμο και δημοφιλές ιδίως σε εφαρμογές χαμηλής κατανάλωσης και μικρού κόστους. Καθώς οι απαιτήσεις των σύγχρονων ενσωματωμένων συστημάτων αυξάνονται το **I2C** προσαρμόζεται διαρκώς διατηρώντας τη θέση του ως ακρογωνιαίο πρωτόκολλο επικοινωνίας, προσφέροντας ιδανική ισορροπία μεταξύ απλότητας, επεκτασιμότητας και απόδοσης για πλήθος εφαρμογών, από τα **smartphones** και τους υπολογιστές μέχρι τα συστήματα αυτοματισμού, τα ιατρικά μηχανήματα και τα οχήματα, καταδεικνύοντας την αντοχή και την προσαρμοστικότητα αυτής της τεχνολογίας στο πέρασμα των δεκαετιών.[52-55]

- Το **SPI** (**Serial Peripheral Interface**) είναι ένα σύγχρονο πρωτόκολλο επικοινωνίας, στο οποίο όλη η διαδικασία ελέγχεται από έναν κεντρικό ελεγκτή. Αναπτύχθηκε αρχικά από τη **Motorola** τη δεκαετία του 1980 και έχει καθιερωθεί ως βασικό πρωτόκολλο διασύνδεσης μικροελεγκτών με περιφερειακές συσκευές σε ενσωματωμένα συστήματα. Πρόκειται για πρωτόκολλο πλήρους διπλής κατεύθυνσης τύπου **master-slave**, που σημαίνει ότι η αποστολή και η λήψη δεδομένων μπορεί να γίνεται ταυτόχρονα. Όταν η συσκευή **master** στέλνει δεδομένα προς τον **slave**, ο **slave** μπορεί ταυτόχρονα να στείλει δεδομένα πίσω στον master, χωρίς να απαιτείται ξεχωριστή εντολή ανάγνωσης. Το **SPI** χρησιμοποιείται ευρέως σε ενσωματωμένα συστήματα, κυρίως σε εφαρμογές που απαιτούν υψηλές ταχύτητες μεταφοράς δεδομένων. Το πρωτόκολλο **SPI** απαιτεί τέσσερις ακίδες: η **MOSI** (**Master Output Slave Input**), η οποία χρησιμοποιείται από τον **master** για αποστολή δεδομένων στον **slave**, η **MISO** (**Master Input Slave Output**), μέσω της οποίας ο **slave** στέλνει δεδομένα στον **master**, η **SCLK** (**Serial Clock**), που παρέχει τον παλμό ρολογιού για συγχρονισμό της μεταφοράς δεδομένων και η **CS** (**Chip Select** ή **Slave Select**), που χρησιμοποιείται από τον **master** για να επιλέξει τη συσκευή **slave** με την οποία θα επικοινωνήσει. Σε επίπεδο υλικού, το **SPI** χρησιμοποιεί συνήθως διαμόρφωση εξόδου **push-pull** για τις ακίδες **SCLK**, **MOSI** και **CS**, πράγμα που σημαίνει ότι μπορούν να οδηγούν τη γραμμή τόσο σε υψηλή όσο και σε χαμηλή στάθμη τάσης. Η ακίδα **MISO** διαμορφώνεται συχνά ως έξοδος ανοικτού συλλέκτη (**open-drain**), πράγμα που σημαίνει ότι μπορεί μόνο να τραβήξει τη γραμμή σε χαμηλή στάθμη και απαιτεί εξωτερική αντίσταση **pull-up** για να επιστρέψει σε υψηλή στάθμη. Αυτό επιτρέπει τη χρήση κοινής γραμμής **MISO** σε πολλαπλές συσκευές **slave**, αφού μόνο μία συσκευή τη χρησιμοποιεί κάθε φορά. Η διαμόρφωση αυτή προστατεύει και από ζημιές σε περίπτωση που δύο συσκευές προσπαθήσουν να οδηγήσουν τη γραμμή ταυτόχρονα. Η ταχύτητα του **SPI** καθορίζεται από τη συχνότητα του ρολογιού, που εξαρτάται από την ταχύτητα του μικροελεγκτή, το μήκος των καλωδίων, το επίπεδο θορύβου στο σύστημα και τον επιθυμητό ρυθμό μετάδοσης δεδομένων. Συνήθως η συχνότητα κυμαίνεται από

λίγα **kHz** έως και αρκετά **MHz**. Όσο πιο γρήγορος είναι ο μικροελεγκτής και όσο μικρότερη είναι η απόσταση μεταξύ **master** και **slave**, τόσο υψηλότερη μπορεί να είναι η συχνότητα ρολογιού και αντίστοιχα η ταχύτητα μεταφοράς δεδομένων. Η διαδικασία μεταφοράς δεδομένων αρχίζει με την επιλογή της συσκευής **slave**, δηλαδή όταν ο **master** "τραβά" τη γραμμή **CS** χαμηλά για να υποδείξει ότι πρόκειται να επικοινωνήσει με αυτήν. Στη συνέχεια, ο **master** ρυθμίζει τις παραμέτρους επικοινωνίας όπως η συχνότητα του ρολογιού, η μορφή των δεδομένων και το μέγεθος της λέξης. Ο **master** στέλνει δεδομένα μέσω της γραμμής **MOSI**, ενώ ταυτόχρονα λαμβάνει δεδομένα από τον **slave** μέσω της γραμμής **MISO**. Η μετάδοση γίνεται **bit** προς **bit**, ξεκινώντας από το πιο σημαντικό **bit** (**MSB**), και συγχρονίζεται από τους παλμούς της **SCLK** που δημιουργεί ο **master**. Όταν ολοκληρωθεί η μεταφορά, ο **master** επαναφέρει τη γραμμή **CS** σε υψηλή στάθμη και ο **slave** σταματά να επικοινωνεί. Σε επίπεδο καταχωρητών, η μεταφορά δεδομένων γίνεται ανά **byte**. Ο **master** στέλνει παλμούς μέσω της **SCLK**, θέτει κατάλληλα το **MOSI** για αποστολή του κάθε **bit** και λαμβάνει μέσω του **MISO** το αντίστοιχο **bit** από τον **slave**. Τα **bits** συσσωρεύονται στα **shift registers** των δύο συσκευών και μετά από 8 κύκλους ρολογιού ολοκληρώνεται η ανταλλαγή ενός **byte**. Το **SPI** υποστηρίζει και τη διαδοχική σύνδεση πολλών συσκευών, γνωστή ως **daisy chaining**. Σε αυτήν τη διάταξη, η έξοδος της μιας συσκευής συνδέεται με την είσοδο της επόμενης και ο **master** ελέγχει την επικοινωνία με ολόκληρη την αλυσίδα μέσω ενός μόνο σήματος **CS**, αποστέλλοντας δεδομένα που προορίζονται για όλες τις συνδεδεμένες συσκευές. Επιπλέον, το **SPI** περιλαμβάνει δυνατότητες όπως ανίχνευση σφαλμάτων μέσω **CRC** (**Cyclic Redundancy Check**), ρύθμιση πολικότητας και φάσης του ρολογιού, που καθορίζουν πότε διαβάζονται και πότε αποστέλλονται τα δεδομένα, προσφέροντας μεγαλύτερη ευελιξία ανάλογα με τις απαιτήσεις της εφαρμογής. Σε σύγκριση με άλλα πρωτόκολλα όπως το **I2C** και το **UART**, το **SPI** προσφέρει αρκετά πλεονεκτήματα. Υποστηρίζει υψηλές ταχύτητες μεταφοράς δεδομένων, έχει απλή δομή με ελάχιστες απαιτήσεις ελέγχου, επιτρέπει πλήρη διπλή επικοινωνία (ταυτόχρονη αποστολή και λήψη) και προσφέρει τη δυνατότητα διαδοχικής σύνδεσης πολλών συσκευών, γεγονός που το καθιστά ιδανικό για εφαρμογές που απαιτούν ταχύτητα και ευελιξία. Η αξιοπιστία και η υψηλή απόδοση του **SPI** το καθιστούν ιδανικό για εφαρμογές που απαιτούν ταχύτατη μετάδοση δεδομένων, όπως η επικοινωνία με εξωτερικές μνήμες **flash**, αισθητήρες υψηλής ακρίβειας, οθόνες **LCD** και **OLED**. Στο πεδίο των ενσωματωμένων συστημάτων και του **Internet of Things** (**IoT**), το **SPI** μπορεί να χρησιμοποιηθεί για την αλληλεπίδραση με περιβαλλοντικούς αισθητήρες για την παρακολούθηση της θερμοκρασίας, της υγρασίας και για άλλες εφαρμογές σε οικιακές συσκευές ή βιομηχανικές εφαρμογές. Επιπλέον, το **SPI** χρησιμοποιείται ευρέως σε ασύρματες μονάδες επικοινωνίας για τη διασύνδεση μικροελεγκτών με ασύρματες συσκευές επικοινωνίας όπως **WiFi** και **Bluetooth**. Αυτές οι συσκευές απαιτούν υψηλές ταχύτητες μετάδοσης για την αποστολή και λήψη δεδομένων χωρίς ενσύρματη.

Συμπερασματικά, η επικοινωνία μέσω **SPI** αποτελεί μία ιδιαίτερα αποδοτική λύση για ενσωματωμένα συστήματα που απαιτούν γρήγορη, αξιόπιστη και απλή μεταφορά δεδομένων. Χάρη στην ευελιξία, την αποδοτικότητα και τη διαδεδομένη χρήση του, το **SPI** παραμένει βασικό εργαλείο στον χώρο των συστημάτων μικροελεγκτών και αναμένεται να συνεχίσει να αποτελεί βασικό πρωτόκολλο και στο μέλλον.[56,57]

- Το **UART** (**Universal Asynchronous Receiver/Transmitter**) αποτελεί ένα βασικό μηχανισμό σειριακής επικοινωνίας μεταξύ δύο συσκευών. Αν και στις καταναλωτικές εφαρμογές έχει αντικατασταθεί από πιο σύγχρονα πρότυπα όπως το **USB**, συνεχίζει να διαδραματίζει σημαντικό ρόλο στον χώρο των ενσωματωμένων συστημάτων (**embedded systems**) και των μικροελεγκτών, όπως σε πλατφόρμες τύπου **Arduino** και **Raspberry Pi**. Η ονομασία του παρέχει βασικές πληροφορίες για τον τρόπο λειτουργίας του: η λέξη "**Universal**" υποδηλώνει τη γενική του χρήση σε διάφορες συσκευές, η λέξη "**Asynchronous**" σημαίνει ότι η επικοινωνία γίνεται χωρίς τη χρήση κοινού σήματος ρολογιού, ενώ οι όροι "**Receiver**" και "**Transmitter**" αναφέρονται στον δέκτη και τον πομπό αντίστοιχα. Έτσι, το **UART** επιτρέπει την αποστολή και τη λήψη δεδομένων χωρίς συγχρονισμό μέσω ρολογιού, αλλά βασίζεται σε κοινά συμφωνημένες παραμέτρους ανάμεσα στις δύο συσκευές. Η λειτουργία του βασίζεται σε δύο μόνο γραμμές: μία γραμμή για τη μετάδοση των δεδομένων (**TX**) και μία για τη λήψη τους (**RX**). Η φυσική σύνδεση των συσκευών είναι απλή, καθώς το **TX** της μίας συσκευής συνδέεται με το **RX** της άλλης και αντίστροφα. Πριν ξεκινήσει η επικοινωνία, οι συσκευές πρέπει να διαμορφωθούν με τις ίδιες παραμέτρους, όπως η ταχύτητα μετάδοσης, το μήκος των δεδομένων και η χρήση ή μη bit ισοτιμίας για τον έλεγχο σφαλμάτων. Καθώς δεν υπάρχει σήμα ρολογιού, η ακριβής ρύθμιση είναι ζωτικής σημασίας, αφού τυχόν απόκλιση στις παραμέτρους μπορεί να προκαλέσει λανθασμένη μετάδοση ή λήψη των δεδομένων. Η επικοινωνία μέσω **UART** γίνεται με αποστολή των δεδομένων σε μορφή πακέτων. Κάθε πακέτο ξεκινά με ένα **start bit** που έχει λογική τιμή 0 για να υποδείξει την αρχή της μετάδοσης. Ακολουθούν τα **bit** των δεδομένων, συνήθως οκτώ, ένα προαιρετικό **parity bit** για έλεγχο σφαλμάτων και ένα ή δύο **stop bits** με τιμή 1 για να δηλώσουν το τέλος του πακέτου. Αυτή η δομή βασίζεται στο πρότυπο **NRZ** (**Non-Return-to-Zero**), όπου το σήμα διατηρεί την τιμή του καθ’ όλη τη διάρκεια του κάθε **bit**. Όταν δεν πραγματοποιείται μετάδοση, η γραμμή παραμένει σε κατάσταση λογικού 1. Το **parity bit** επιτρέπει την ανίχνευση απλών σφαλμάτων. Για παράδειγμα, αν οριστεί άρτια ισοτιμία και στα δεδομένα υπάρχουν τέσσερα **bit** με τιμή 1, τότε το **parity bit** τίθεται σε 0 ώστε το συνολικό πλήθος των "1" να είναι ζυγός. Αν όμως κατά τη μετάδοση αλλάξει κάποιο **bit** λόγω παρεμβολών και το πλήθος γίνει περιττό, η λήπτρια συσκευή εντοπίζει την ασυμφωνία και αντιλαμβάνεται την ύπαρξη σφάλματος. Ένα σημαντικό χαρακτηριστικό του **UART** είναι η πλήρης αμφίδρομη επικοινωνία (**full-duplex**), καθώς οι συσκευές μπορούν

να στέλνουν και να λαμβάνουν δεδομένα ταυτόχρονα. Το πρωτόκολλο είναι εύκολο στη διαμόρφωση και δεν απαιτεί πολύπλοκο υλικό, καθώς χρησιμοποιεί μόνο δύο καλώδια. Επιπλέον, η απουσία ρολογιού απλοποιεί τον σχεδιασμό, αλλά ταυτόχρονα καθιστά απαραίτητο τον ακριβή συγχρονισμό των δύο πλευρών μέσω των προκαθορισμένων παραμέτρων. Η σειριακή επικοινωνία, όπως αυτή του **UART**, υπερτερεί σε απλότητα και ευκολία υλοποίησης σε σχέση με την παράλληλη, η οποία απαιτεί πολλαπλές γραμμές και σύνθετη καλωδίωση. Ωστόσο, η παράλληλη επικοινωνία προσφέρει σημαντικά υψηλότερες ταχύτητες. Για τον λόγο αυτό, το **UART** χρησιμοποιείται κυρίως όταν η απλότητα και η αξιοπιστία είναι πιο σημαντικές από την απόδοση, όπως σε αισθητήρες, μονάδες **Bluetooth**, **GPS** και μικροελεγκτές. Παρά τα πλεονεκτήματά του, το **UART** έχει και ορισμένους περιορισμούς. Η απουσία ρολογιού το καθιστά ευάλωτο σε χρονικές αποκλίσεις. Επίσης, τα επιπλέον **bits** που συνοδεύουν τα δεδομένα, όπως το **start**, το **stop** και το **parity bit**, μειώνουν την καθαρή απόδοση της επικοινωνίας, δημιουργώντας ένα είδος επιβάρυνσης. Επιπλέον, το **UART** δεν είναι κατάλληλο για επικοινωνία πολλαπλών συσκευών, όπως άλλες τεχνολογίες τύπου **SPI** ή **I2C**. Συνοψίζοντας, το **UART** είναι ένας βασικός πυλώνας στην ψηφιακή επικοινωνία, του οποίου η κατανόηση είναι απαραίτητη για οποιονδήποτε ασχολείται με μικροελεγκτές και γενικότερα με συστήματα ενσωματωμένου ελέγχου. Προσφέρει έναν σταθερό και αποδοτικό τρόπο μετάδοσης δεδομένων με ελάχιστες απαιτήσεις σε υλικό και λογισμικό, ενώ η ευρεία του χρήση σε πληθώρα εφαρμογών το καθιστά αναπόσπαστο κομμάτι κάθε σύγχρονης ηλεκτρονικής σχεδίασης.[58,59]

5.5 Διαχείριση μνήμης και βελτιστοποίηση κώδικα

Σε αντίθεση με τις παραδοσιακές γλώσσες όπως την **C/C++**, όπου η ευθύνη της διαχείρισης μνήμης ανήκει αποκλειστικά στον προγραμματιστή, η **MicroPython** εφαρμόζει έναν πιο εξελιγμένο και αυτοματοποιημένο μηχανισμό. Η αυτόματη διαχείριση μνήμης απαλλάσσει τον προγραμματιστή από τη χειροκίνητη διαχείριση, προσφέροντας ταυτόχρονα μεγαλύτερη ασφάλεια και σταθερότητα στις εφαρμογές. Αυτή η προσέγγιση εξαλείφει δύο από τα πιο συνηθισμένα προβλήματα της χειροκίνητης διαχείρισης μνήμης: τις διαρροές μνήμης (**memory leaks**), όπου η μνήμη δεν απελευθερώνεται ποτέ, και την πρόσβαση σε ήδη αποδεσμευμένη μνήμη (**dangling pointers**), που μπορεί να προκαλέσει απρόβλεπτη συμπεριφορά. Στον πυρήνα αυτής της αυτοματοποιημένης διαχείρισης βρίσκεται ο μηχανισμός **Garbage Collector**, ο οποίος είναι υπεύθυνος για τη συνεχή παρακολούθηση της μνήμης και την απελευθέρωση χώρου που δεν χρησιμοποιείται. Όταν ένα αντικείμενο δεν είναι πλέον προσβάσιμο από κανένα σημείο του προγράμματος, ο **Garbage Collector** το εντοπίζει και αποδεσμεύει αυτόματα τη μνήμη που το καταλάμβανε, διασφαλίζοντας έτσι την βέλτιστη αξιοποίηση των διαθέσιμων πόρων.[60]

Η **MicroPython** ενσωματώνει σημαντικές βελτιστοποιήσεις με στόχο τη μέγιστη αποδοτικότητα σε συστήματα με περιορισμένους πόρους. Οι τεχνικές αυτές εστιάζουν κυρίως στη διαχείριση **bytecode**, τη βελτιστοποίηση μεταβλητών και τις στρατηγικές διαχείρισης μνήμης. Στον τομέα της διαχείρισης **bytecode**, αξιοποιείται το **frozen bytecode**, το οποίο επιτρέπει την εκτέλεση του κώδικα απευθείας από τη μνήμη **ROM**, μειώνοντας έτσι την κατανάλωση **RAM** και βελτιώνοντας την συνοχή της μνήμης σωρού. Επιπλέον, η χρήση **cross compiler** για την δημιουργία αρχείων τύπου **.mpy** ενισχύει σημαντικά την χρήση αποθηκευτικού χώρου. Για την βελτιστοποίηση των μεταβλητών, η **MicroPython** εφαρμόζει τεχνικές όπως ο διαχωρισμός τοπικών και καθολικών μεταβλητών για την επιτάχυνση της πρόσβασης και την εξοικονόμηση μνήμης. Η χρήση της συνάρτησης **const()** για τον ορισμό σταθερών τιμών συμβάλλει στη μείωση των απαιτήσεων μνήμης, ενώ η επιλογή σύντομων ονομάτων μεταβλητών περιορίζει την κατανάλωση **RAM**. Σε επίπεδο διαχείρισης μνήμης, βασικές δομές δεδομένων τοποθετούνται κατά προτίμηση στη στοίβα αντί στον σωρό, επιτυγχάνοντας ταχύτερη πρόσβαση και πιο αποδοτική χρήση των διαθέσιμων πόρων. Αντίθετα, για δυναμικές δομές και εισαγωγές δεδομένων απαιτείται χώρος στον σωρό. Επίσης, η αρχική δήλωση καθολικών μεταβλητών δεσμεύει τμήμα της μνήμης, γεγονός που επηρεάζει τον προγραμματισμό σε περιορισμένα συστήματα.[61]

ΚΕΦΑΛΑΙΟ 6 Δημιουργία και Προσομοίωση MicroPython Εφαρμογών στο Wokwi

6.1 Εισαγωγή στο Wokwi και πλεονεκτήματα χρήσης

Το **Wokwi** αποτελεί ένα σύγχρονο εργαλείο προσομοίωσης ηλεκτρονικών συστημάτων και μικροελεγκτών, διαθέσιμο μέσω διαδικτύου, το οποίο επιτρέπει τη δοκιμή εφαρμογών και τον πειραματισμό χωρίς την ανάγκη φυσικού υλικού. Υποστηρίζει μια ευρεία γκάμα μικροελεγκτών, όπως το **Arduino**, το **ESP32** και το **STM32**, καθώς και πλήθος περιφερειακών συσκευών και αισθητήρων.

Το βασικό πλεονέκτημα του **Wokwi** έγκειται στη δυνατότητα λειτουργίας σε εικονικό περιβάλλον, όπου τα σφάλματα δεν επιφέρουν καταστροφικές συνέπειες. Οι χρήστες έχουν την δυνατότητα να πειραματίζονται ελεύθερα, με την επιλογή επαναφοράς αλλαγών ανά πάσα στιγμή. Αυτή η προσέγγιση προσφέρει σημαντικό εκπαιδευτικό όφελος, ιδιαίτερα για αρχαρίους στον τομέα των ενσωματωμένων συστημάτων. Η πλατφόρμα διευκολύνει τη συνεργασία και την ανταλλαγή ιδεών μέσω της δυνατότητας κοινής χρήσης έργων, καθώς οι χρήστες μπορούν να ζητούν βοήθεια ή να λαμβάνουν σχόλια απλά μοιράζοντας έναν σύνδεσμο του έργου τους. Επιπλέον, η εικονική φύση του περιβάλλοντος εξαλείφει τους περιορισμούς που σχετίζονται με τη φυσική διαθεσιμότητα εξαρτημάτων, επιτρέποντας τη δημιουργία πολύπλοκων συστημάτων χωρίς κόστος ή περιορισμούς αποθέματος. Ακόμα, η πλατφόρμα προσφέρει [62]:

- Πλήρη προσομοίωση δικτυακών πρωτοκόλλων (**WiFi, MQTT, HTTP, NTP**).
- Εικονικός λογικός αναλυτής για παρακολούθηση ψηφιακών σημάτων (**UART, I2C, SPI**).
- Ενσωματωμένο εργαλείο εντοπισμού σφαλμάτων (**GDB**) για προχωρημένους χρήστες.
- Προσομοίωση συσκευών αποθήκευσης (κάρτες **SD**) με δυνατότητα διαχείρισης αρχείων.
- Δυνατότητα δημιουργίας προσαρμοσμένων εξαρτημάτων (**Chips API**) και ενσωμάτωσης με το **Visual Studio Code**.

6.2 Προγραμματισμός LED Blink στο Wokwi

Ο παρακάτω αποτελεί βασικό παράδειγμα προγραμματισμού για το αναβόσβημα (**blink**) ενός **LED** στην πλατφόρμα **Arduino**, όπως υλοποιείται στο περιβάλλον προσομοίωσης **Wokwi**:

```
// Η συνάρτηση setup εκτελείται μία φορά κατά την εκκίνηση ή επανεκίννηση της πλακέτας
void setup() {
  // Αρχικοποίηση του ψηφιακού αποδέκτη LED_BUILTIN ως έξοδος
  pinMode(LED_BUILTIN, OUTPUT);
}

// Η συνάρτηση loop εκτελείται συνεχώς σε ατέρμονα βρόχο
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // ανάβει το LED (HIGH αντιστοιχεί σε ενεργό)
  delay(1000);                       // καθυστερηση 1 δευτερόλεπτο (1000ms = 1sec)
  digitalWrite(LED_BUILTIN, LOW);    // σβήνει το LED (LOW αντιστοιχεί σε μηδενική)
  delay(1000);                       // καθυστέρηση 1 δευτερόλεπτο
}
```

Εικόνα 5: Το LED είναι συνδεδεμένο στον ακροδέκτη 13 της Arduino μέσω μίας αντίστασης 220Ω

Ο κώδικας ελέγχει το ενσωματωμένο **LED** της πλακέτας **Arduino**, αναβοσβήνοντας το με χρονικό διάστημα ενός δευτερολέπτου (1 δευτερόλεπτο ενεργοποιημένο, 1 δευτερόλεπτο απενεργοποιημένο) σε συνεχή επανάληψη. Οι περισσότερες πλακέτες **Arduino** διαθέτουν ενσωματωμένο **LED**, το οποίο στα μοντέλα **UNO**, **MEGA** και **ZERO**, είναι συνδεδεμένο στον ψηφιακό ακροδέκτη 13, ενώ στο μοντέλο **MKR1000** βρίσκεται στον ακροδέκτη 6. Η σταθερά **LED_BUILTIN** χρησιμοποιείται για την αυτόματη αναφορά στον σωστό ακροδέκτη ανάλογα με το εκάστοτε μοντέλο πλακέτας, διευκολύνοντας έτσι την φορητότητά του κώδικα μεταξύ διαφορετικών πλατφορμών.[63]

6.3 Προσομοίωση αισθητήρων και διασύνδεση με MicroPython

Το **Wokwi** υποστηρίζει μια μεγάλη ποικιλία εικονικών αισθητήρων, όπως θερμοκρασίας, υγρασίας, κίνησης και φωτεινότητας, δίνοντας στους χρήστες τη δυνατότητα να πειραματιστούν με διαφορετικές σενάρια και συνθήκες λειτουργίας.[64]

- **HC-SR04** (Υπερηχητικός Αισθητήρας Απόστασης)

Pin	Περιγραφή	Λειτουργία
VCC	Παρέχει τάση λειτουργίας 5V	Τροφοδοσία
TRIG	Εκπέμπει παλμό για ενεργοποίηση υπερήχων	Έναρξη μέτρησης
ECHO	Διάρκεια HIGH παλμού ανάλογη με απόσταση	Λήψη αποτελεσμάτων
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση

```
#define PIN_TRIG 3 // Ορίζει τον ακροδέκτη 3 ως Trig (εκκίνηση μέτρησης)
#define PIN_ECHO 2 //Ορίζει τον ακροδέκρη 2 ως Echo (λήψη αποτελεσμάτων)
//Αρχικοποίση setup
void setup() {
  Serial.begin(115200);      // Εκκίνηση σειριακής επικοινωνίας
  pinMode(PIN_TRIG, OUTPUT); // Ρύθμιση  TRIG ως έξοδος
  pinMode(PIN_ECHO, INPUT);  // Ρύθμιση ECHO ως είσοδος
}
//Συνάρτηση loop, εκτελείται συνεχώς
void loop() {
  // Εκκίνηση νέας μέτρησης:
  digitalWrite(PIN_TRIG, HIGH); // Ενεργοποίηση TRIG
  delayMicroseconds(10);        // Παλμός 10ms (καθυστέρηση)
  digitalWrite(PIN_TRIG, LOW);  // Απενεργοποίηση TRIG

  // Ανάγνωση αποτελεσμάτων:
  int duration = pulseIn(PIN_ECHO, HIGH); //Μέτρηση διάρκεις παλμού ECHO (ms)
  // Εμφάνιση αποτελεσμάτων:
  Serial.print("Distance in CM: ");
  Serial.println(duration / 58); // Μετατροπή σε cm (1cm = 58ms)
  Serial.print("Distance in inches: ");
  Serial.println(duration / 148); // Μεταρτροπή σε ίντσες (1 ίντσα = 148ms)

  delay(1000); // Καθυστέρηση 1 δευτερολέπτου ανάμεσα στις μετρήσεις
}
```

Εικόνα 6: Παράδειγμα σε Arduino (https://wokwi.com/projects/304444938977804866)

- **DHT22** (Αισθητήρας Υγρασίας και Θερμοκρασίας)

Pin	Περιγραφή	Λειτουργία
VCC	Θετική τάση τροφοδοσίας	Τροφοδοσία
SDA	Ψηφιακή είσοδος/έξοδος δεδομένων	Μονόδρομη επικοινωνία δεδομένων
NC	Μη συνδεδεμένος (Not Connected)	Δεν χρησιμοποιείται
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση

```ino
#include "DHTesp.h"

const int DHT_PIN = 15; //Ορισμός του ακροδέκτη για τον αισθητήτα

DHTesp dhtSensor; //Δημιουργία αντικειμένου αισθητήρα

void setup() {
  Serial.begin(115200); // Εκκίνηση σειριακής επικοινωνίας
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22); // Ρύθμιση του αισθητήρα DT22
}

void loop() {
  TempAndHumidity  data = dhtSensor.getTempAndHumidity(); // Ανάγνωση δεδομένων
  Serial.println("Temp: " + String(data.temperature, 2) + "°C"); // Εκτύπωση θερμοκρασίας
  Serial.println("Humidity: " + String(data.humidity, 1) + "%"); // Εκτύπωση υγρασίας
  Serial.println("---"); // Διαχωριστική γραμμή
  delay(2000); //  Καθυστέρηση 2 δευτερολέπτων
}
```

Εικόνα 7: DHT22 στον ESP32 (https://wokwi.com/projects/344892337559700051)

- **DS1307 RTC** (Ενότητα Ρολογιού Πραγματικού Χρόνου [**RTC**] με διεπαφή **I2C** και 56 **bytes** μη πτητικής στατικής μνήμης **RAM**)

Pin	Περιγραφή	Λειτουργία
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση
5V	Χρησιμοποιείται για την τροφοδοσία της συσκευής	Θετική Τάση
SDA	Υπεύθυνη για τη μεταφορά των δεδομένων στην επικοινωνία I2C	Γραμμή δεδομένων I2C
SCL	Συγχρονίζει τη μεταφορά δεδομένων στην επικοινωνία I2C	Γραμμή ρολογιού I2C
SQW	Μπορεί να παράγει σήματα διαφόρων συχνοτήτων	Έξοδος τετραγωνικού κύματος

```ino
#include "RTClib.h"

RTC_DS1307 rtc; //Δημιουργία αντικειμένου για το RTC

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

void setup () {
  Serial.begin(115200); // Εκκίνηση σειριακής επικοινωνίας

  if (! rtc.begin()) { // Προσπάθεια αρχικοποίησης RTC
    Serial.println("Couldn't find RTC"); // Μήνυμα σφάλματος
    Serial.flush();
    abort(); // Τερματισμός προγράμματος σε περίπτωση αποτυχίας
  }
}
```

```
void loop () {
  DateTime now = rtc.now(); // Ανάγνωση τρέχουσας ημρομηνίας/ώρας
// Εκτύπωση ημερομηνίας
  Serial.print("Current time: ");
  Serial.print(now.year(), DEC); // Έτος
  Serial.print('/');
  Serial.print(now.month(), DEC); // Μήνας
  Serial.print('/');
  Serial.print(now.day(), DEC); // Ημέρες
  Serial.print(" (");
  Serial.print(daysOfTheWeek[now.dayOfTheWeek()]); // Εκτύπωση ημέρας εβδομάδας
  Serial.print(") ");
  Serial.print(now.hour(), DEC); // Ώρες
  Serial.print(':');
  Serial.print(now.minute(), DEC); // Λεπτά
  Serial.print(':');
  Serial.print(now.second(), DEC); // Δευτερόλεπτα
  Serial.println();


  Serial.println();
  delay(3000); // Καθυστέρηση 3 δευτερολέπτων
}
```

Εικόνα 8: Διάβασμα τωρινής ημερομηνίας/ώρας με χρήση RTClib (https://wokwi.com/projects/305979285237137984)

- **PIR Motion Sensor** (Αισθητήρας Κίνησης Παθητικού Υπερύθρου)

Pin	Περιγραφή	Λειτουργία
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση
OUT	Ψηφιακή έξοδος σήματος	Έξοδος (Output)
VCC	Θετική τάση τροφοδοσίας	Τροφοδοσία

```
int ledPin = 13;                // Ορισμός ακροδέκτη για το LED
int inputPin = 2;               // Ορισμός ακροδέκτη εισόδου (για PIR αισθητήρα)
int pirState = LOW;             // Αρχική κατάσταση, δεν ανιχνεύεται κίνηση
int val = 0;                    // Μεταβλητή για ανάγνωση κατάστασης ακροδέκτη

void setup() {
  pinMode(ledPin, OUTPUT);      // Ορισμός LED ως έξοδος
  pinMode(inputPin, INPUT);     // Ορισμός αισθητήρα ως είσοδος

  Serial.begin(9600); // Εκκίνηση σειριακής επικοινωνίας
}
```

```
void loop() {
  val = digitalRead(inputPin);  // Ανάγνωση τιμής από τον αισθητήρα
  if (val == HIGH) {            // Έλεγχος αν ανιχνεύεται κίνηση (HIGH)
    digitalWrite(ledPin, HIGH);  // Άναμμα LED
    if (pirState == LOW) {
      // Εκτύπωση μηνύματος μόνο κατά την αλλαγή κατάστασης
      Serial.println("Motion detected!");
      // Εκτύπωση μηνύματος μόνο κατά την αλλαγή κατάστασης
      pirState = HIGH;
    }
  } else {
    digitalWrite(ledPin, LOW); // Σβήσιμο LED
    if (pirState == HIGH) {
      // we have just turned of
      Serial.println("Motion ended!");
      // Εκτύπωση μηνύματος μόνο κατά την αλλαγή κατάστασης
      pirState = LOW;
    }
  }
}
```

Εικόνα 9: Παράδειγμα PIR αισθητήρας (AdaFruit), (https://wokwi.com/projects/299284655047180808)

- **Analog Temperature Sensor** (**NTC**) (Αναλογικός Αισθητήρας Θερμοκρασίας)

Pin	Περιγραφή	Λειτουργία
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση
OUT	Ψηφιακή έξοδος σήματος	Έξοδος (Output)
VCC	Θετική τάση τροφοδοσίας	Τροφοδοσία

```
const float BETA = 3950; // Πρέπει να ταιριάζει με τον συντελεστή Beta του θερμίστορ

void setup() {
  Serial.begin(9600); // Εκκίνηση σειριακής επικοινωνίας
}

void loop() {
  int analogValue = analogRead(A0); // Ανάγνωση αναλογικής τιμής από τον ακροδέκτη
  // Υπολογισμός θερμοκρασίας σε βαθμούς Κελσίου
  float celsius = 1 / (log(1 / (1023. / analogValue - 1)) / BETA + 1.0 / 298.15) - 273.15;
  // Εκτύπωση αποτελεσμάτων
  Serial.print("Temperature: ");
  Serial.print(celsius);
  Serial.println(" °C");
  delay(1000); Καθυστέρηση 1 δευτερολέπτου ανάμεσα σε μετρήσεις
}
```

Εικόνα 10: Παράδειγμα NTC θερμίστορ (https://wokwi.com/projects/299330254810382858)

- **DS18B20 Temperature Sensor** (Ψηφιακός Αισθητήρας Θερμοκρασίας Μονής Γραμμής)

- **MPU6050** (Ολοκληρωμένος αισθητήρας με τριαξονικό επιταχυνσιόμετρο, τριαξονικό γυροσκόπιο και αισθητήρα θερμοκρασίας με διεπαφή **I2C**)

Pin	Περιγραφή	Λειτουργία
VCC	Θετική τάση τροφοδοσίας	Τροφοδοσία
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση
SCL	Συγχρονίζει τη μεταφορά δεδομένων στην επικοινωνία I2C	Γραμμή ρολογιού I2C
SDA	Υπεύθυνη για τη μεταφορά των δεδομένων στην επικοινωνία I2C	Γραμμή δεδομένων I2C
XDA	Εφεδρική γραμμή δεδομένων για πρόσθετες διεπαφές	Δευτερεύουσα γραμμή δεδομένων
XCL	Εφεδρική γραμμή ρολογιού για πρόσθετες διεπαφές	Δευτερεύουσα γραμμή ρολογιού
AD0	Ρύθμιση διεύθυνσης I2C (LOW/HIGH)	Επιλογή διεύθυνσης συσκευής
INT	Σήμα διακοπής (Interrupt)	Έξοδος διακοπής για ασύγχρονη ειδοποίηση μικροελεγκτή

```ino
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu; // Δημιουργία αντικειμένου για τον αισθητήρα MPU6050

void setup(void) {
  Serial.begin(115200); // Εκκίνηση σειριακής επικοινωνίας

  while (!mpu.begin()) {
    Serial.println("MPU6050 not connected!");
    delay(1000); //Αναμονή 1 δευτερολέπτου
  }
  Serial.println("MPU6050 ready!");
}

sensors_event_t event; // Δομή για αποθήκευση των μετρήσεων

void loop() {
  // Ανάγνωση των μετρήσεων επιτάχυνσης
  mpu.getAccelerometerSensor()->getEvent(&event);
  // Εκτύπωση των αποτελεσμάτων
  Serial.print("[");
  Serial.print(millis()); // Χρόνος από την εκκίνηση (ms)
  Serial.print("] X: ");
  Serial.print(event.acceleration.x); // Επιτάχυνση στον άξονα X
  Serial.print(", Y: ");
  Serial.print(event.acceleration.y); // Επιτάχυνση στον άξονα Y
  Serial.print(", Z: ");
  Serial.print(event.acceleration.z); // Επιτάχυνση στον άξονα Z
  Serial.println(" m/s^2"); // Μονάδες μέτρησης (μέτρα ανά δευτερόλεπτο τετράγωνο)
  delay(500); // Αναμονή 500ms ανάμεσα σε μετρήσεις
}
```

Εικόνα 11: Παράδειγμα σε Arduino (https://wokwi.com/projects/305937248748044864)

- **Photoresistor** (Αισθητήρας Φωτοαντίστασης (**LDR**))

Pin	Περιγραφή	Λειτουργία
VCC	Θετική τάση τροφοδοσίας	Τροφοδοσία
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση
DO	Ψηφιακή έξοδος (Digital Output)	Έξοδος HIGH/LOW
AO	Αναλογική έξοδος (Analog Output)	Έξοδος αναλογικής τάσης

```ino
#include <LiquidCrystal_I2C.h> // Βιβλιοθήκη για οθόνη LCD μέσω I2C

#define LDR_PIN 2 // Ορισμός ακροδέκτη για τον αισθητήρα φωτός (LDR)

LiquidCrystal_I2C lcd(0x27, 20, 4); // Δημιουργία αντικειμένου LCD (διεύθυνση I2C 0x27, 20x4 χαρακτήρες)

void setup() {
  pinMode(LDR_PIN, INPUT); // Ορισμός ακροδέκτη LDR ως είσοδος
  lcd.init();              // Αρχικοποίηση οθόνης LCD
  lcd.backlight();         // Ενεργοποίηση φωτισμού οθόνης
}

void loop() {
  lcd.setCursor(2, 0); // Θέση κέρσορα (στήλη 2, γραμμή 0)
  lcd.print("Room: "); // Εκτύπωση στατικού κειμένου
  if (digitalRead(LDR_PIN) == LOW) { // Έλεγχος κατάστασης LDR
    lcd.print("Light!"); // Αν LOW (φωτεινό)
  } else {
    lcd.print("Dark  "); // Αν HIGH (σκοτεινό)
  }
  delay(100); // Καθυστέρηση 100ms ανάμεσα σε ενημερώσεις
}
```

Εικόνα 12: Ψηφιακό Παράδειγμα Χρήσης Φωτοαντίστασης (https://wokwi.com/projects/305193592908939842)

- **HX711 Load Cell** (Ενισχυτής κελιού φορτίου **HX711** με κύτταρο μέτρησης **5kg**/**50kg**/εκατοστόμετρο)

Pin	Περιγραφή	Λειτουργία
VCC	Θετική τάση τροφοδοσίας	Τροφοδοσία
DT	Σειριακά δεδομένα (Data)	Έξοδος δεδομένων από τον ADC (Digital Output)
SCK	Σειριακό ρολόι (Clock)	Σήμα ρολογιού για συγχρονισμό επικοινωνίας
GND	Σύνδεση με κοινό δυναμικό αναφοράς (0V) του συστήματος. Απαραίτητο για κλείσιμο κυκλώματος	Γείωση

```ino
#include "HX711.h" // Βιβλιοθήκη για ενισχυτή HX711

HX711 scale; // Δημιουργία αντικειμένου για τη ζυγαρία

void setup() {
  Serial.begin(9600); // Εκκίνηση σειριακής επικοινωνίας
  Serial.println("Initializing the scale"); // Μήνυμα έναρξης
  scale.begin(A1, A0); // Αρχικοποίηση ζυγαρίας(DT σε A1, SCK σε A0)
}

void loop() {
  Serial.println(scale.get_units(), 1); // Εκτύπωση βάρους με 1 δεκαδικό ψηφίο
  delay(1000); // Καθυστέρηση 1 δευτερολέπτου ανάμεσα σε μετρήσεις
}
```

Εικόνα 13: Παράδειγμα σε Arduino (https://wokwi.com/projects/345134808605655636)

ΚΕΦΑΛΑΙΟ 7 Συμπεράσματα και Μελλοντικές Προεκτάσεις

7.1 Αξιολόγηση της MicroPython για ενσωματωμένα συστήματα

Η **MicroPython** εισάγει μια νέα εποχή στον προγραμματισμό ενσωματωμένων συστημάτων, προσφέροντας μοναδικά πλεονεκτήματα έναντι των παραδοσιακών γλωσσών. Σε αντίθεση με τη δυσνόητη σύνταξη της **C**, η **MicroPython** διαθέτει καθαρή και ευανάγνωστη δομή, καθώς υιοθετεί την φιλοσοφία της Python για ευκολία μάθησης και ταχεία ανάπτυξη, ιδανική για γρήγορα πρωτότυπα και ευέλικτη συντήρηση κώδικα. Η **MicroPython** προσφέρει έτοιμες δομές **try/except** και **try/finally**, επιτρέποντας την έξυπνη διαχείριση σφαλμάτων με αποτέλεσμα την κομψή αντιμετώπιση εξαιρέσεων χωρίς διακοπή της λειτουργίας του συστήματος. Ως έργο ανοιχτού κώδικα υπό την άδεια **MIT**, η **MicroPython** παρέχει πλήρη ελευθερία χρήσης και τροποποίησης. Οι προγραμματιστές μπορούν εύκολα να την προσαρμόσουν σε διάφορους μικροελεγκτές, ενώ η δυνατότητα συνεισφοράς στην κοινότητα ενισχύει συνεχώς τις δυνατότητες της. Η **MicroPython** φέρνει τεχνικές αντικειμενοστραφούς προγραμματισμού (κληρονομικότητα, πολυμορφισμός) στον κόσμο των ενσωματωμένων συστημάτων. Παρά τις θεωρητικές δυνατότητες υλοποίησης τέτοιων δομών, η **MicroPython** τις κάνει εφικτές και ασφαλείς με ελάχιστη προσπάθεια. Με την εισαγωγή της βιβλιοθήκης **pyb**, η **MicroPython** αποτελεί δραστικά την αλληλεπίδραση με το υλικό. Η διαχείριση περιφερειακών συστημάτων όπως τα **LED** γίνεται μέσω απλών μεθόδων, απελευθερώνοντας τους μηχανικούς από τις χαμηλού επιπέδου λεπτομέρειες και επιτρέποντας τους να επικεντρωθούν στη λογική της εφαρμογής. Παρά τους περιορισμούς της σε κρίσιμες εφαρμογές πραγματικού χρόνου, η ευκολία χρήσης και οι σύγχρονές δυνατότητες της την καθιστούν ένα ισχυρό εργαλείο στον μηχανισμό των ενσωματωμένων συστημάτων.[65]

7.2 Προκλήσεις και περιορισμοί

Η φύση της **MicroPython** εισάγει σημαντικές προκλήσεις στην προστασία του κώδικα, καθώς οι διαθέσιμες επιλογές αποθήκευσης (είτε ως απλά **Python scripts** είτε ως μεταγλωττισμένα **bytecode** αρχεία) δεν προσφέρουν ισχυρή ασφάλεια. Αν και το **bytecode** καθιστά την ανάγνωση του κώδικα λίγο πιο δύσκολη, δεν εμποδίζει αποτελεσματικά την επαναφορά του σε αναγνώσιμη μορφή. Επομένως, οι προγραμματιστές οφείλουν να αξιολογούν προσεκτικά το απαιτούμενο επίπεδο προστασίας και, όπου είναι απαραίτητο, να εφαρμόζουν πρόσθετα μέτρα για τη διασφάλιση της πνευματικής τους ιδιοκτησίας. Παράλληλα, η αξιοπιστία του συστήματος σε περίπτωση σφαλμάτων αποτελεί κρίσιμο ζήτημα. Ο κώδικας της εφαρμογής μπορεί να είναι αποθηκευμένος είτε απευθείας στον μικροελεγκτή είτε σε εξωτερική μνήμη, όπως κάρτα **SD**, ανάλογα με τη σχεδίαση του προϊόντος. Έχει διαπιστωθεί ότι το σύστημα αρχείων της **MicroPython** είναι ευάλωτο σε ξαφνικές διακοπές ρεύματος ή πτώσεις τάσης, με αποτέλεσμα την πιθανή καταστροφή του. Σε τέτοιες περιπτώσεις, η **MicroPython** προσπαθεί να επαναφέρει το σύστημα αντιγράφοντας μια προκαθορισμένη εικόνα στο σύστημα αρχείων. Για να είναι

αποτελεσματική αυτή η διαδικασία, οι προγραμματιστές πρέπει να φροντίσουν ώστε ο βασικός κώδικας να είναι ενσωματωμένος στο **firmware**, δίνοντας τη δυνατότητα στο σύστημα να επανέλθει στις εργοστασιακές ρυθμίσεις και να εφαρμόσει τυχόν διαθέσιμες ενημερώσεις από άλλες περιοχές μνήμης. Επιπλέον, η διαχείριση της μνήμης παίζει σημαντικό ρόλο στην αξιοπιστία ενός τελικού προϊόντος. Η σειρά **pyboard D**, η οποία αποτελεί την πιο πρόσφατη αναπτυξιακή πλατφόρμα της **MicroPython**, περιλαμβάνει δύο ανεξάρτητες **SPI** μνήμες των 2 **MB** η καθεμία. Η πρώτη χρησιμοποιείται για την αποθήκευση της εφαρμογής, ενώ η δεύτερη μπορεί να περιέχει δεδομένα ή άλλες κρίσιμες πληροφορίες. Η λογική αυτή μπορεί να λειτουργήσει ως πρότυπο για παραγωγικά συστήματα, τα οποία ενδείκνυται να διαθέτουν επιπλέον αποθηκευτικούς χώρους και αντίγραφα ασφαλείας του **firmware**, ώστε να εξασφαλίζεται η ομαλή αποκατάσταση της εφαρμογής σε περίπτωση βλάβης, χωρίς επιπτώσεις στον τελικό χρήστη.[66]

7.3 Μελλοντικές εξελίξεις και πιθανές βελτιώσεις

Η έκδοση **MicroPython** 2.0 σηματοδοτεί μια σημαντική μετάβαση, καθώς μετά από σχεδόν δέκα χρόνια σταθερότητας στη σειρά **1.x**, εισάγονται αλλαγές που διακόπτουν τη συμβατότητα, με στόχο τη βελτίωση της συνοχής, της απόδοσης και της επεκτασιμότητας του έργου. Η μετάβαση δεν αποτελεί απλή αναβάθμιση, αλλά στρατηγική επαναξιολόγηση βασικών στοιχείων του **API**, με σκοπό τη δημιουργία μιας ευέλικτης και ενιαίας πλατφόρμας. Αν και δεν υπάρχουν ακόμη επίσημες εκδόσεις υλικολογισμικού, οι χρήστες μπορούν να μεταγλωττίσουν την έκδοση ενεργοποιώντας τη ρύθμιση **MICROPY_PREVIEW_VERSION_2**. Στο υλικό, στόχος είναι η ενοποίηση του **module machine** σε όλες τις πλατφόρμες, ώστε να διευκολυνθεί η συγγραφή επαναχρησιμοποιήσιμου κώδικα και τεκμηρίωσης ανεξαρτήτως μικροελεγκτή. Σε επίπεδο λειτουργικού συστήματος, η **MicroPython** 2.0 υποστηρίζει την εκτέλεση **.mpy** αρχείων απευθείας από το σύστημα αρχείων, μειώνοντας την κατακερματισμένη χρήση **RAM** και επιταχύνοντας την ανάπτυξη. Προωθείται επίσης ευελιξία στη διαμόρφωση αποθηκευτικών μέσων και υποστήριξη για **USB mass storage**. Όσον αφορά τη συμβατότητα με την **CPython**, αφαιρούνται **MicroPython-specific** επεκτάσεις από βασικά **modules**. Για παράδειγμα, συναρτήσεις και κλάσεις όπως **os.mount** και **os.VfsFat** μεταφέρθηκαν στο νέο **module vfs**, διευκολύνοντας τη μεταφορά και δοκιμή κώδικα μεταξύ **CPython** και **MicroPython**. Η **MicroPython** 2.0 αποτελεί ορόσημο για την κοινότητα των ενσωματωμένων συστημάτων. Παρά τις αρχικές απαιτήσεις προσαρμογής, οι βελτιώσεις που εισάγει εξασφαλίζουν ισχυρότερη συμβατότητα, καλύτερη διαχείριση πόρων και μακροπρόθεσμη βιωσιμότητα των έργων.

Στον πυρήνα της ψηφιακής εποχής, τα ενσωματωμένα συστήματα αποτελούν τον αφανή αλλά ζωτικής σημασίας κινητήριο μοχλό της τεχνολογικής προόδου. Από τις έξυπνες φορητές συσκευές μέχρι τους εξελιγμένους βιομηχανικούς αυτοματισμούς, αυτά τα μικροσκοπικά αλλά ισχυρά υπολογιστικά υποσυστήματα έχουν διεισδύσει σχεδόν σε κάθε πτυχή της σύγχρονης ζωής, ενισχύοντας τη λειτουργικότητα, την αποδοτικότητα και την ασφάλεια ποικίλων εφαρμογών. Η συνεχής εξέλιξή τους αποτελεί έναν από τους πιο εντυπωσιακούς και δυναμικούς τεχνολογικούς άξονες, καθώς πλέον συνδυάζονται στενά με το Διαδίκτυο των Πραγμάτων (**IoT**) και την Τεχνητή Νοημοσύνη (**AI**), δημιουργώντας ένα νέο πρότυπο διασυνδεδεμένων, ευφυών συστημάτων. Αυτά τα συστήματα είναι σε θέση

να βελτιστοποιούν την ενεργειακή κατανάλωση, να ενισχύουν την κυβερνοασφάλεια μέσω προηγμένων μηχανισμών κρυπτογράφησης, να υποστηρίζουν σύγχρονες υποδομές **cloud** και **mesh** δικτύωσης, να εφαρμόζουν τεχνικές βαθιάς μάθησης για προσαρμοζόμενες λύσεις και να προσφέρουν διαδραστική οπτικοποίηση δεδομένων σε πραγματικό χρόνο. Οι προβλέψεις της αγοράς αντικατοπτρίζουν τη ραγδαία άνοδο του τομέα, σύμφωνα με τη μελέτη της **QYResearch**, η αξία της παγκόσμιας αγοράς ενσωματωμένων συστημάτων, που το 2017 ανερχόταν σε 68,9 δισεκατομμύρια δολάρια, προβλέπεται να φτάσει τα 105,7 δισεκατομμύρια έως το 2025. Η ανάπτυξη αυτή δεν περιορίζεται μόνο στο οικονομικό επίπεδο, αλλά σηματοδοτεί μια βαθιά ποιοτική μεταμόρφωση στον τρόπο με τον οποίο ζούμε, εργαζόμαστε και παράγουμε, με τα ενσωματωμένα συστήματα να παίζουν καθοριστικό ρόλο στην ψηφιακή μετεξέλιξη της κοινωνίας και της βιομηχανίας.[6,67,68]

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.ibm.com/think/topics/internet-of-things

[2]https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence

[3]https://www.tutorchase.com/answers/ib/computer-science/how-can-embedded-systems-impact-society

[4] https://bluefruit.co.uk/processes/a-brief-history-of-embedded-operating-systems/

[5]https://www.heavy.ai/technical-glossary/embedded-systems

[6]chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.dit.uoi.gr/e-class/modules/document/file.php/158/%CE%94%CE%B9%CE%AC%CE%BB%CE%B5%CE%BE%CE%B7/EMB_systems_intro.pdf

[7]https://www.spiceworks.com/tech/tech-general/articles/what-are-embedded-systems/

[8]https://www.geeksforgeeks.org/real-time-operating-system-rtos/

[9]https://www.spiceworks.com/tech/tech-general/articles/what-is-assembly-language/#_001

[10]https://www.geeksforgeeks.org/introduction-of-assembler/

[11]https://codefinity.com/blog/Assembler-Programming

[12]https://www.geeksforgeeks.org/what-is-assembly-language/

[13]https://www.investopedia.com/terms/a/assembly-language.asp

[14]https://www.geeksforgeeks.org/c-programming-language/

[15]https://www.coursera.org/articles/c-programming

[16]https://www.totalphase.com/blog/2019/12/basics-embedded-c-programming/?srsltid=AfmBOorAp2GMnvqVEZxrMhbyOiGjlpuougf4dMMaLQ_lBEdBatiK6tlz

[17]https://shorturl.at/KKCJa

[18]https://shorturl.at/l0XMx

[19]https://unstop.com/blog/history-of-cpp

[20]https://shorturl.at/lAQZ2

[21]https://www.geeksforgeeks.org/dynamic-programming/

[22]https://www.geeksforgeeks.org/introduction-to-dynamic-programming-data-structures-and-algorithm-tutorials/

[23]https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/

[24]https://www.digikey.com/en/maker/projects/micropython-basics-what-is-micropython/1f60afd88e6b44c0beb0784063f664fc

[25]https://www.kevsrobots.com/blog/what-is-micropython.html

[26]https://shorturl.at/rEW2Z

[27]https://www.ko2.co.uk/micropython-vs-python/

[28]https://www.youtube.com/watch?v=urVVE0LQdvY

[29]https://blog.grobotronics.com/?p=1026

[30] https://www.nabto.com/guide-to-iot-esp-32/

[31] https://www.espressif.com/en/products/socs/esp32

[32]https://peer.asee.org/raspberry-pi-pico-as-an-iot-device

[33]https://community.element14.com/products/raspberry-pi/b/blog/posts/raspberry-pi-s-rp2040-in-house-chip-brings-high-performance-at-low-cost---inspiration-abounds

[34]https://www.sparkfun.com/raspberry-pi-pico.html

[35]https://www.digikey.gr/en/product-highlight/s/stmicroelectronics/stm32-overview

[36]https://www.ampheo.com/blog/comparison-of-fpga-arm-stm32-and-dsp-platforms?srsltid=AfmBOorble3XNnPj3ymEeAhjrFdIX4-iNzGTDwkj5-Qzvn1OIuqEh1jc

[37] https://medium.com/pythoneers/micropython-ides-and-tools-aebe61b0f0c5

[38]https://www.linkedin.com/pulse/article-thonny-python-ide-dr-tharanitharan-gurusamy-ilpwc

[39]https://lemariva.com/blog/2018/12/micropython-visual-studio-code-as-ide

[40]https://www.dfrobot.com/blog-710.html?srsltid=AfmBOooiW5Y9b9K3X6iBaiO3Df6S_4ToYvSU7RnN2ifir8Yt4LSVrozy

[41]https://github.com/dhylands/rshell

[42]https://docs.micropython.org/en/latest/reference/pyboard.py.html

[43]https://pypi.org/project/adafruit-ampy/

[44]https://www.lenovo.com/in/en/glossary/putty/?orgRef=https%253A%252F%252Fwww.google.com%252F#

[45]https://thonny.org/

[46] https://docs.pycom.io/firmwareapi/pycom/machine/pin/

[47]https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide?srsltid=AfmBOorKMpM7U649_85Oq6p-wnaXrOk0slQk4ssQ2dWE99PzNzHgIxIg#

[48] https://learn.adafruit.com/micropython-basics-blink-a-led/blink-led

[49]https://www.electromaker.io/project/view/blink-a-led-with-esp32-and-micropython?srsltid=AfmBOorSzCRcDF0RixwroXyyDlRetwN00TkTfreNQ13wrlAKk5kEa7jY

[50]chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://halvorsen.blog/documents/technology/iot/pico/resources/Raspberry%20Pi%20Pico%20and%20Push%20Buttons.pdf

[51]https://www.youtube.com/watch?v=4AaaaUr7z2c&list=PL2935W76vRNEV6PwvxiqJf47JVrbGzTkC&index=3

[52] https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[53] https://www.embedded.com/introduction-to-i2c/

[54] https://www.geeksforgeeks.org/i2c-communication-protocol/

[55] https://learn.sparkfun.com/tutorials/i2c/a-brief-history-of-i2c

[56]https://embeddedwala.com/Blogs/DigitalCommunication/Getting-Started-with-SPI:-A-Beginners-Guide

[57] https://www.geeksforgeeks.org/what-is-serial-peripheral-interface-spi/

[58] https://www.circuitbasics.com/basics-uart-communication/

[59]https://www.geeksforgeeks.org/universal-asynchronous-receiver-transmitter-uart-protocol/

[60] https://docs.micropython.org/en/latest/develop/memorymgt.html

[61] https://docs.micropython.org/en/latest/develop/optimizations.html

[62] https://docs.wokwi.com/

[63] https://wokwi.com/projects/344891652101374548

[64] https://docs.wokwi.com/getting-started/supported-hardware

[65]https://www.beningo.com/5-advantages-of-using-micro-python-for-embedded-software/

[66]https://www.designnews.com/testing-measurement/the-pros-and-cons-of-designing-embedded-systems-with-micropython

[67] https://docs.micropython.org/en/latest/reference/micropython2_migration.html

[68]https://www.linkedin.com/pulse/unveiling-role-embedded-systems-our-daily-lives-cpp-psp-pmp-pci-cc-go7mf