
**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

# «Μελέτη και αξιολόγηση κυρίαρχων περιβαλλόντων ανάπτυξης αλγορίθμων για ενσωματωμένα συστήματα με έμφαση στην πειραματική αποτίμηση της απόδοσης και κατανάλωσης μιας ARM αναπτυξιακής πλατφόρμας»

Αθανάσιος Θανόπουλος Α.Μ. 0982

Αντώνιος Σιτοκωνσταντίνου Α.Μ. 1093

ΕΠΙΒΛΕΠΩΝ**:** Νικόλαος Βώρος, Επίκουρος Καθηγητής

ΕΠΙΒΛΕΠΩΝ: Χρήστος Αντωνόπουλος, Επιστημονικός Συνεργάτης

ΝΑΥΠΑΚΤΟΣ 2015

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Ναύπακτος,

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. ΒΩΡΟΣ ΝΙΚΟΛΑΟΣ

2. ΑΝΤΩΝΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ

3. ΑΣΑΡΙΔΗΣ ΗΛΙΑΣ

## ΕΙΣΑΓΩΓΗ

Η σχεδίαση ψηφιακών συστημάτων έχει εισάγει μια νέα εποχή. Σε μια εποχή που ο σχεδιασμός μικροεπεξεργαστών έχει εξελιχθεί σε μία κλασική άσκηση βελτιστοποίησης, η σχεδίαση ενσωματωμένων υπολογιστικών συστημάτων, στα οποία οι μικροεπεξεργαστές αποτελούν μόνο ένα τμήμα τους, αποτελεί ανοιχτή πρόκληση. Ασύρματα συστήματα, συστήματα σε ρούχα, δικτυωμένα συστήματα, έξυπνες συσκευές, συστήματα βιομηχανικών διεργασιών, προηγμένα συστήματα αυτοκινήτων και βιολογικά διασυνδεδεμένα συστήματα αποτελούν μερικά μόνο παραδείγματα της νέας εποχής που εχει ανατείλει.

## ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Ενσωματωμένα συστήματα, αρχιτεκτονική ARM, αρχιτεκτονικές RISC-CISC, SoC, FPGA, Cortex M0+, αποτίμηση απόδοσης και κατανάλωσης ισχύος ARM επεξεργαστών, λειτουργικά συστήματα, δίκτυα αισθητήρων

# ΚΕΦΑΛΑΙΟ 1$^{Ο}$

## 1.1 ΠΕΡΙΛΗΨΗ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Στην παρούσα εργασία αρχικά θα γίνει αποτίμηση της απόδοσης και της κατανάλωσης ισχύος της τελευταίας γενιάς των επεξεργαστών τύπου ARM. Αυτοί οι επεξεργαστές υπάρχουν στην πλειοψηφία των κινητών τηλεφώνων και tablets. Πιο συγκεκριμένα, αρχικά θα αναπτυχθεί ένα περιβάλλον καταγραφής των μετρήσεων της απόδοσης / ισχύος και στη συνέχεια θα γίνει προσπάθεια μοντελοποίησης των χαρακτηριστικών του ARM επεξεργαστή με βάση τα αποτελέσματα των μετρήσεων.

## 1.2 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

Η ραγδαία ανάπτυξη των υπολογιστικών συστημάτων έχει επηρεάσει κυρίως την κατηγορία των "μικρο-υπολογιστών". Αυτό έχει ως αποτέλεσμα οι διάφορες υποκατηγορίες όπως οι προσωπικοί υπολογιστές (PC) και τα ενσωματωμένα συστήματα να τείνουν όλο και περισσότερο να συγχωνευτούν μεταξύ τους. Έτσι, όσο κυλάει ο χρόνος γίνεται όλο και πιο δύσκολο να ξεχωρίσει κανείς τους προσωπικούς υπολογιστές από τα ενσωματωμένα συστήματα. Μέχρι πριν λίγα χρόνια, ενσωματωμένο σύστημα θεωρούταν οποιοδήποτε σύστημα εκτελούσε μία μόνο λειτουργία, 'οπως για παράδειγμα ο έλεγχος της θερμοκρασίας σε ένα φούρνο, ο έλεγχος τη ταχύτητας ενός κινητήρα κ.τ.λ. Σήμερα όμως, υπάρχουν συσκευές όπως έξυπνα κινητά(smartphones), PDA's(personal digital assistans) κ.α., τα οποία έχουν την δυνατότητα να εκτελούν πολλές λειτουργίες μαζί ανήκοντας όμως στην κατηγορία των ενσωματωμένων συστημάτων. Έχουν παραμείνει σ΄αυτή την κατηγορία και δεν έχουν συγχωνευτεί ακόμα με την κατηγορία των προσωπικών υπολογιστών(PC), γιατί τουλάχιστον μέχρι σήμερα, χρησιμοποιούν περιορισμένο υλικό(hardware), μεθόδους μείωσης της κατανάλωσης ενέργειας(λειτουργία με μπαταρίες), καθώς και ειδικό λογισμικό(firmware). Έτσι, ένα ενσωματωμένο υπολογιστικό σύστημα εξυπηρετεί μία ή περισσότερες εφαρμογές οι οποίες δεν ξεπερνούν σε απαιτήσεις επεξεργαστικής ισχύ έναν προσωπικό υπολογιστή.

Ας δούμε μερικά παραδείγματα ενσωματωμένων εφαρμογών, κατηγοριοποιημένα. Ο κατάλογος είναι μόνο ενδεικτικός, για να εκτιμηθεί το εύρος της αγοράς αλλά και οι 3 δυνατότητες για περαιτέρω ανάπτυξη τέτοιων συστημάτων. Σε όλες τις παρακάτω κατηγορίες συστημάτων υπάρχει επεξεργαστής, στον οποίο ο χρήστης δεν έχει άμεση πρόσβαση. Για παράδειγμα, μπορεί σε ψηφιακά παιχνίδια τύπου Gameboy να μπορεί ο/η χρήστης να επιλέξει παιχνίδι με εισαγωγή κατάλληλης κασέτας, αλλά δεν έχει την δυνατότητα να προγραμματίσει την κονσόλα. Επίσης, σε διάφορα συστήματα όπως δρομολογητές δικτύων η κατασκευάστρια εταιρία μπορεί να αλλάζει το πρόγραμμά τους, αλλά και πάλι ο χρήστης δεν έχει άμεση πρόσβαση στο λογισμικό. Ενδεικτικές κατηγορίες είναι:

Υπολογιστές και Περιφερειακά

- Ασύρματα περιφερειακά (π.χ. Bluetooth ακουστικά/μικρόφωνα, IR ποντίκια και πληκτρολόγια, κλπ.)
- Ασύρματα δίκτυα (routers και κάρτες για WiFi, 802.11, Bluetooth, κλπ.)
- Κονσόλες παιχνιδιών (π.χ. Sony Playstation, Microsoft Xbox, κλπ.)

Είδη Προσωπικής Ευκολίας

- Φορητά Παιχνίδια (π.χ. Gameboy, Nintendo, κλπ.)
- Κινητά τηλέφωνα
- Προσωπικοί Ψηφιακοί Βοηθοί (PDA)
- Φορητά συστήματα παγκοσμίου εντοπισμού (GPS – Global Positioning Systems)

Αυτοκίνητα

- Συστήματα ελέγχου απόδοσης μηχανής (καθορισμό μίγματος, χρονισμό, κλπ.)
- Συστήματα ελέγχου ρύπων
- Συστήματα ελέγχου άνεσης καμπίνας επιβατών (π.χ. κλιματισμός, ρυθμίσεις καθισμάτων, ρυθμίσεις καθρεφτών, κλπ.)

Οικιακές Συσκευές

- Ψυγεία, κουζίνες, φούρνοι μικροκυμάτων
- Τηλεοράσεις, συσκευές εικόνας (Video Cassette recorder – VCR, Digital Video Disc - DVD)
- Στερεοφωνικά νέας γενιάς και συστήματα αιθουσών προβολής σπιτιού (Home Theater)

Βιομηχανικά Συστήματα

- Βιομηχανικά Ρομπότ
- Αριθμητικά ελεγχόμενα μηχανουργικά μηχανήματα (π.χ. φρέζες με αριθμητικό έλεγχο – Computer Numerical Control – CNC)

Υγεία, και Υποστηρικτική Τεχνολογία για Άτομα με Ειδικές Ανάγκες (AMEA)

- Φορητοί καρδιογράφοι
- Συστήματα καθαρισμού αίματος
- Συστήματα παρακολούθησης ζωτικών λειτουργιών ασθενών
- Απηνιδοτές
- Αναπηρικά αμαξίδια
- Συσκευές εισόδου ελεγχόμενες από επιστόμιο
- Συστήματα δημιουργίας ήχου (σύνθεση φωνής)

Τηλεπικοινωνίες

- Ψηφιακά τηλεφωνικά κέντρα
- Δικτυακός εξοπλισμός (δρομολογητές - routers, μεταγωγείς - switches, κλπ.)
- Δορυφορικά συστήματα

Αγροτική Παραγωγή και Περιβάλλον

- Συστήματα παρακολούθησης και ελέγχου συνθηκών εδάφους
- Συστήματα ελέγχου περιβάλλοντος και αυτόματου ταΐσματος σε κτηνοτροφικές μονάδες
- Συστήματα παρακολούθησης ρύπων

- Συστήματα έγκαιρης προειδοποίησης φυσικών καταστροφών

Ασφάλεια

- Συστήματα παρακολούθησης σπιτιών και συναγερμοί
- Συστήματα πυρόσβεσης

Μεταφορές

- Ηλεκτρονικά αεροσκαφών (avionics)
- Συστήματα εντοπισμού θέσης λεωφορείων, τραίνων, κλπ. και

ενημέρωσης επιβατών για επικείμενες αφίξεις

- Φωτεινές ενδείξεις χρόνου αναμονής σε σηματοδότες οδικής

κυκλοφορίας και κυκλοφορίας πεζών

- Συστήματα για αυτόματη παρακολούθηση θέσης γραμμάτων/δεμάτων σε ταχυμεταφορικές εταιρίες

Υπηρεσίες

- Συστήματα αυτόματων τραπεζικών συναλλαγών (ATM)
- Συστήματα διατήρησης προτεραιότητας ουρών (π.χ. σε τράπεζες)
- Φωτεινές ενδείξεις (π.χ. κυλιόμενες οθόνες)
- Φορητά συστήματα για παραγγελίες σε εστιατόρια, κλπ.

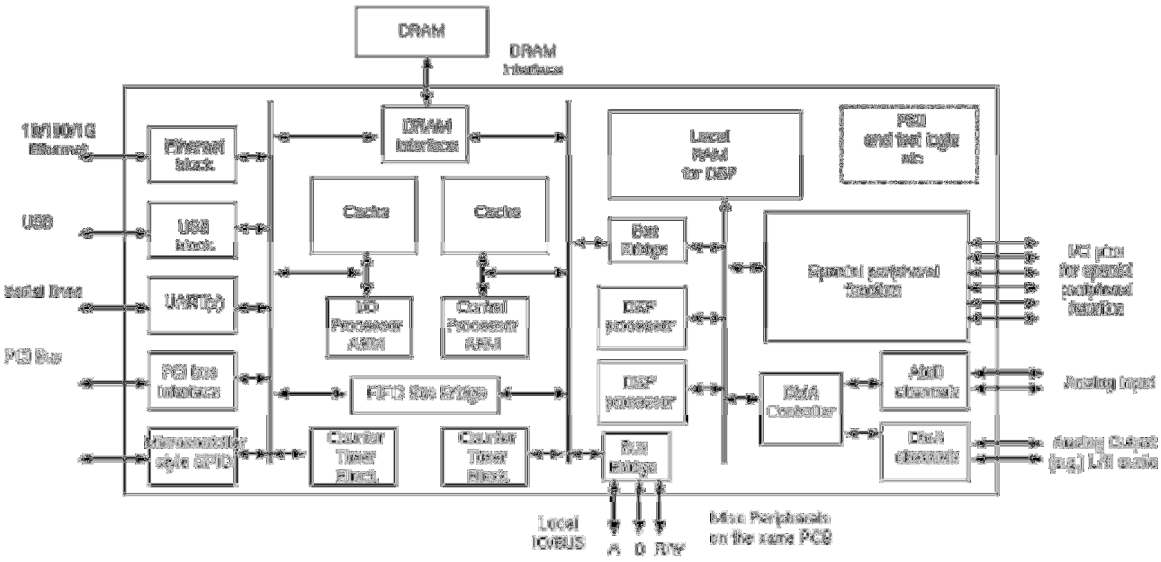
Δημόσια Διοίκηση

- Αυτόματα συστήματα στάθμευσης
- Συστήματα για κλήσεις σε παραβάτες κυκλοφορίας, στάθμευσης, κλπ

# ΚΕΦΑΛΑΙΟ 2$^{Ο}$ - ΟΡΙΣΜΟΙ ΒΑΣΙΚΩΝ ΕΝΝΟΙΩΝ

## 2.1 ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

### 2.1.1 System On a Chip (SoC)




**ΕΙΚΟΝΑ 1 : BLOCK ΔΙΑΓΡΑΜΜΑ ΕΝΟΣ SOC**

Η τεχνολογία System on a Chip αφορά στο συνδυασμό όλων εκείνων των απαραίτητων ηλεκτρονικών κυκλωμάτων που αποτελούν μέρος ενός συστήματος και βρίσκονται ενσωματωμένα σε ένα και μόνο κύκλωμα, ευρέως γνωστό και ως microchip (Greaves, 2011). Για παράδειγμα ένα system on a chip για μία συσκευή ανίχνευσης ήχου θα μπορούσε να περιλαμβάνει έναν λήπτη, έναν μετατροπέα αναλογικού σε ψηφιακό, έναν μικροεπεξεργαστή, μία μνήμη αλλά και έναν ελεγκτή εισόδου/εξόδου σε ένα και μόνο microchip (Greaves, 2011).

Η τεχνολογία αυτή χρησιμοποιείται σε μικρές και ιδιαίτερα σύνθετες, καταναλωτικές ηλεκτρονικές συσκευές. Κάποιες μάλιστα από αυτές τις συσκευές έχουν ιδιαίτερα υψηλή επεξεργαστική δύναμη αλλά και μνήμη μεγαλύτερη από ένα σταθερό υπολογιστή 10 ετών (Greaves, 2011). Στο μέλλον αναμένεται ότι η SoC τεχνολογία θα ενσωματωθεί στα nanorobots και θα λειτουργεί με τρόπο τέτοιο ώστε να αντιμετωπίσει τις έως τώρα ανίατες ασθένειες. Οι SoC βίντεο – συσκευές μπορεί να

ενσωματωθούν στον ανθρώπινο εγκέφαλο και για παράδειγμα να βοηθήσουν έναν αριθμό τυφλών ανθρώπων ώστε να ξαναδεί ή κάποιους από τους κωφούς ανθρώπους ώστε να ξανακούσουν (Greaves, 2011). Επίσης, μικροί υπολογιστές χειρός με μικρά τσιπ κεραιών θα έχουν τη δυνατότητα στο άμεσο μέλλον να κάνουν αναζήτηση στο ίντερνετ με ταχύτητες της τάξεως των gigabyte/second από οποιοδήποτε μέρος της γης.

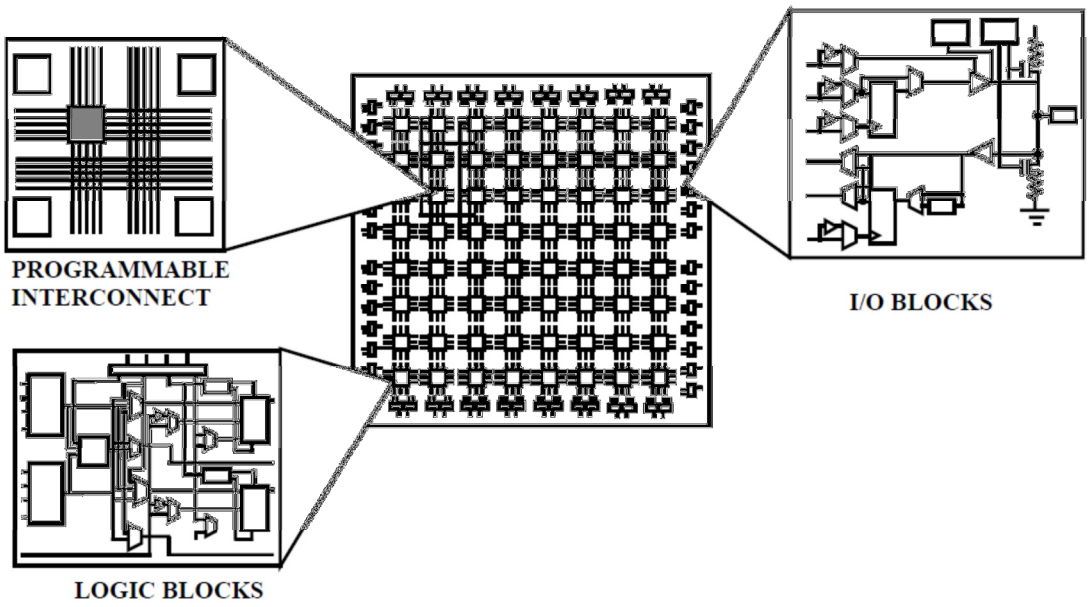
Κάθε SoC όπως προειπώθηκε αποτελεί ένα ολοκληρωμένο σύστημα σε ένα τσιπ. Ένα σύστημα περιλαμβάνει, έναν μικροεπεξεργαστή, μία μνήμη και διάφορα περιφερειακά. Ο επεξεργαστής μπορεί να είναι κάποιος από τους γνωστούς μικροεπεξεργαστές, κάποιος από αυτούς που αφορούν σε πολυμέσα και ήχο ή κάποιες από τις σύγχρονες εφαρμογές βίντεο (Wikipedia). Οι τύποι των επεξεργαστών μπορεί να είναι πολλοί και διάφοροι και να συνδέονται μεταξύ τους διαμέσου διαφόρων μηχανισμών που συμπεριλαμβάνουν διαμοιραζόμενες μνήμες και οντότητες διανομής μηνυμάτων όπως για παράδειγμα συγκεκριμένα κανάλια. Τα SoC υπάρχουν πλέον σχεδόν σε κάθε καταναλωτικό προϊόν όπως τα κινητά τηλέφωνα, τα DVDs, τα modems, οι τηλεοράσεις και τα iPODs (Wikipedia).

Παρακάτω παρουσιάζονται αναλυτικά τα μέρη από τα οποία αποτελείται ένα SoC (Wikipedia) :

- Μικροεπεξεργαστής
- Rom, Ram, Eeprom και flash μνήμη
- Περιφερειακά όπως μετρητές, μετρητές πραγματικού χρόνου και γεννήτριες ισχύος
- Αναλογικές διεπαφές
- Ρυθμιστές τάσεις και κυκλώματα διαχείρισης ισχύος

### 2.1.2 Field Progammable Gate Array (FPGA)

Ένα FPGA είναι μία συσκευή η οποία περιέχει ένα κύκλωμα πινάκων επαναπροσδιοριζόμενων λογικών πυλών. Κάθε φορά που διαμορφώνεται ένα FPGA, η σύνδεση του εσωτερικού κυκλώματος γίνεται με τρόπο τέτοιο ώστε η ανάπτυξη του υλικού να είναι ανάλογη της εφαρμογής του λογισμικού (Field Programmable Gate Arrays and Applications ). Αντίθετα με τους επεξεργαστές, οι FPGAs χρησιμοποιούν συγκεκριμένο υλικό σε ότι αφορά τη λογική επεξεργασία και δεν έχουν λειτουργικό σύστημα. Η φύση τους είναι πραγματικά παράλληλη και έτσι οι διάφορες επεξεργαστικές λειτουργίες δεν χρειάζονται να ανταγωνίζονται για τις ίδιες πηγές (Field Programmable Gate Arrays and Applications ). Αποτέλεσμα αυτού είναι αφενός να μην επηρεάζεται η εφαρμογή όταν προστίθεται επιπλέον επεξεργασία και αφετέρου να εκτελούνται πολλαπλοί βρόγχοι σε ένα και μόνο FPGA σε διαφορετικούς ρυθμούς.




**ΕΙΚΟΝΑ 2 : ΕΣΩΤΕΡΙΚΗ ΔΟΜΗ ΕΝΟΣ FPGA**

Σε αντίθεση με τα κυκλώματα PCB, που έχουν συγκεκριμένο υλικό τροφοδοσίας, τα FPGA συστήματα μπορούν να αναδιαρθρώσουν των εσωτερικό τους κύκλωμα και να προωθήσουν την αναδιαμόρφωση τους αφού το σύστημα έχει αναπτυχθεί στο χώρο.

Οι FPGA συσκευές αφορούν κυρίως στην απόδοση και την αξιοπιστία ενός κυκλώματος. Μία και μόνο FPGA συσκευή μπορεί να αντικαταστήσει χιλιάδες διαφορετικών στοιχείων συναθροίζοντας εκατομμύρια λογικών πυλών σε ένα μόνο ενσωματωμένο κύκλωμα ενός chip (Wikipedia). Τα εσωτερικά μέρη ενός FPGA τσιπ αποτελούνται από έναν πίνακα λογικών πυλών που περικλείονται από μία περιφερειακή I/O.

Τα σήματα δρομολογούνται μέσα στον πίνακα από διάφορους διασυνδεδεμένους διακόπτες και καλώδια. Τα λογικά block υλοποιούνται με χρήση πολλαπλών χαμηλού επιπέδου fan in πυλών, που οδηγούν σε ένα πολύ πιο συμπαγή σχεδιασμό σε σχέση πάντα με την υλοποίηση ενός κυκλώματος AND – OR. Κάθε FPGA δίνει στο χρήστη τη δυνατότητα να διαμορφώσει (Field Programmable Gate Arrays and Applications ) :

1. Το σημείο ένωσης των λογικών block
2. Τη συνάρτηση για κάθε ένα από τα block

Κάθε λογικό block μιας FPGA μπορεί να διαμορφωθεί με τρόπο τέτοιο ώστε να παρέχει την ίδια λειτουργικότητα είτε ενός τρανζίστορ είτε ενός μικροεπεξεργαστή. Μπορεί να χρησιμοποιηθεί ώστε να υλοποιήσει διαφορετικούς συνδυασμούς συνδυαστικών και ακολουθιακών λογικών συναρτήσεων (Wikipedia). Τα λογικά block σε μία FPGA μπορούν να υλοποιηθούν με ένα από τους παρακάτω τρόπους (Field Programmable Gate Arrays and Applications ) :

- Ζεύγη τρανζίστορ
- Συνδυασμός λογικών πυλών
- Πίνακες αναζήτησης n-εισόδων
- Πολυπλέκτες
- Ευρεία fan-in AND-OR αρχιτεκτονική

Η δρομολόγηση σε ένα FPGA αποτελείται από τμήματα καλωδίου διαφορετικών μηκών που μπορούν να διασυνδεδεθούν διαμέσου ηλεκτρικών προγραμματιζόμενων διακοπτών. Η πυκνότητα ενός λογικού block που χρησιμοποιείται σε μια FPGA αποτελείται από το μήκος και τον αριθμό των τμημάτων του καλωδίου που χρησιμοποιούνται για τη δρομολόγηση (Wikipedia). Ο αριθμός των τμημάτων καλωδίου που χρησιμοποιούνται για την διασύνδεση ουσιαστικά αποτελεί ένα trade

off ανάμεσα στις λογικές πύλες και στην περιοχή που χρησιμοποιείται για τη δρομολόγηση.

Για ποιο λόγο όμως είναι πλέον χρήσιμοι οι FPGAs ? Στις αρχές του 1980, τα ενσωματωμένα κυκλώματα μικρής κλίμακας σχημάτιζαν το back bone των περισσότερων λογικών κυκλωμάτων στα συστήματα. Οι μικροεπεξεργαστές, οι ελεγκτές I/O, μετρητές και πολλά άλλα υλοποιούνταν με χρήση της τεχνολογίας των ενσωματωμένων κυκλωμάτων (Field Programmable Gate Arrays and Applications ). Οι ενδιάμεσες συνδέσεις χρησιμοποιούνταν ακόμη ώστε να βοηθήσουν στη σύνδεση ανάμεσα στα μεγάλα ηλεκτρονικά κυκλώματα και να τα βοηθήσουν να (Field Programmable Gate Arrays and Applications ):

1. Παράγουν σήματα παγκόσμιου ελέγχου
2. Μεταδίδουν σήματα από ένα υποσύστημα σε ένα άλλο

Τα συστήματα ουσιαστικά αποτελούνταν από μερικά ενσωματωμένα στοιχεία μεγάλης κλίμακας και ένα μεγάλο αριθμό μικρών ενσωματωμένων κυκλωμάτων. Η αρχική προσπάθεια επίλυσης του προβλήματος αυτού οδήγησε στην ανάπτυξη κυκλωμάτων τέτοιων, ανάλογα με τις απαιτήσεις της κάθε περίστασης που είχαν ως στόχο την αντικατάσταση του μεγάλου αριθμού των διασυνδέσεων (Wikipedia). Η μειωμένη αυτή πολυπλοκότητα συστήματος αλλά και το κόστος κατασκευής βελτίωσαν σημαντικά την απόδοση. Βέβαια και τα κυκλώματα αυτά παρουσίαζαν τα μειονεκτήματά τους καθώς ήταν ιδιαίτερα ακριβά στο να αναπτυχθούν και καθυστερούσαν σημαντικά να φθάσουν στην αγορά λόγω του απαιτητικού σχεδιασμού τους. Έτσι λοιπόν η λύση αυτή ήταν βιώσιμη μόνο σε περιπτώσεις όπου υπήρχε μεγάλος όγκος παραγωγής και χαμηλός χρόνος απόκρισης στην αγορά.

Οι FPGAs παρουσιάστηκαν ως μία εναλλακτική υλοποίησης ενός ολόκληρου συστήματος σε ένα μόνο τσιπ αλλά και ως λύση που παρήγαγε ευελιξία του επαναπρογραμματισμού στο χρήστη (Wikipedia). Η εισαγωγή τους οδήγησε στη σημαντική βελτίωση της πυκνότητας που είναι σχετική με τον αριθμό των μικρών ανεξάρτητων κυκλωμάτων.

Ένα άλλο επίσης σημαντικό πλεονέκτημα των FPGA

είναι ότι με τη βοήθεια σχεδιασμού από υπολογιστή μπορούν να υλοποιηθούν σε πολύ μικρό χρονικό διάστημα.

## 2.2 ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

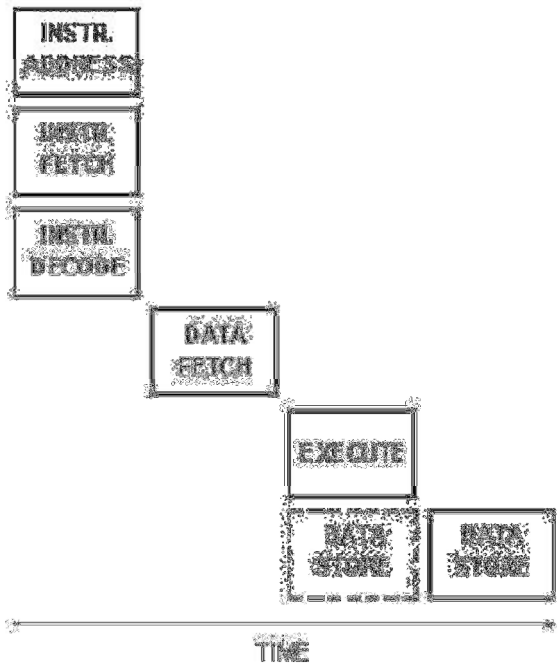
### 2.2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ RISC ΚΑΙ CISC

Η κυρίαρχη αρχιτεκτονική στο χώρο της αγοράς των υπολογιστών ανήκει στο σχεδιασμό CISC (Complex Instruction Set Computer). Ο προφανής λόγος του χαρακτηρισμού complex – σύνθετη αφορά κυρίως στην ιδιαίτερα σύνθετη και πολύπλοκη φύση των αρχιτεκτονικών. Το κίνητρο για το σχεδιασμό τέτοιων αρχιτεκτονικών είναι η παροχή μιας αρχιτεκτονικής τέτοιας που θα υποστηρίζει όλες τις λειτουργίες αλλά και τις δομές δεδομένων που υποστηρίζονται από γλώσσες υψηλότερου επιπέδου (Katevenis). Βέβαια, όπως σε κάθε περίπτωση έτσι και σε αυτόν τον σχεδιασμό υπάρχουν και παράπλευρες επιρροές.

Η απόφαση των σχεδιαστών των CISC επεξεργαστών να παρέχουν μία πληθώρα τρόπων λειτουργίας οδήγησε την ίδια στιγμή σε εντολές μεταβλητού μεγέθους (Masood). Για παράδειγμα, το μέγεθος των εντολών μεγαλώνει εάν ένας τελεστής βρίσκεται στη μνήμη και όχι σε έναν καταχωρητή. Ο λόγος για τον οποίο συμβαίνει αυτό οφείλεται στο ότι η διεύθυνση της μνήμης πρέπει να οριστεί ως μέρος της κωδικοποίησης, γεγονός το οποίο απαιτεί περισσότερα bits (Masood). Όλο αυτό όπως είναι αντιληπτό δυσχεραίνει τη διαδικασία της αποκωδικοποίησης και του προγραμματισμού. Μία άλλη παράπλευρη επιρροή που αφορά στο μεγάλο εύρος των τύπων εντολών είναι ότι ο αριθμός των ρολογιών που χρειάζονται για να εκτελεστούν όλες οι εντολές ποικίλει σημαντικά. Αυτό με τη σειρά του οδηγεί σε προβλήματα τόσο στον προγραμματισμό όσο και στο pipeline.

Για όλους τους παραπάνω λόγους, στις αρχές της δεκαετίας του 1980 οι σχεδιαστές ξεκίνησαν να αναζητούν απλούστερες αρχιτεκτονικές. Λόγω του ότι οι αρχιτεκτονικές αυτές περιείχαν πολύ μικρότερα σετ εντολών οι σχεδιαστές επινόησαν τον όρο RISC (Reduced Instruction Set Computer) (Masood). Παρόλο που ο βασικός στόχος δεν ήταν η μείωση του αριθμού των εντολών αλλά της πολυπλοκότητας ο όρος παρέμεινε.

Ακριβώς ορισμός για το τι ακριβώς περιλαμβάνει ο σχεδιασμός αυτός δεν υπάρχει. Υπάρχει όμως μία περιγραφή κάποιων βασικών χαρακτηριστικών που τείνουν να εμφανίζονται στα συστήματα αυτά.



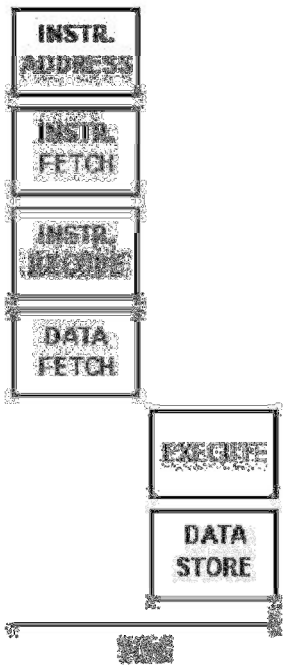

**ΕΙΚΟΝΑ 3 : ΕΣΩΤΕΡΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ RISC**

Η αρχιτεκτονική RISC βασίζεται σε έναν μικροεπεξεργαστή που κάνει χρήση μικρών, ιδιαίτερα βελτιστοποιημένων εντολών, όχι ιδιαίτερα εξειδικευμένων όπως συναντάται σε άλλες αρχιτεκτονικές. Τα πρώτα RISC projects ξεκίνησαν από την IBM, το Stanford και το UC-Berkeley στα τέλη της δεκαετίας του 70 και στις αρχές της δεκαετίας του 80 (Masood). Η 801της IBM, η 3 MIPS του Stanford και η RISC 1&2 του Berkeley σχεδιάστηκαν όλες με παρόμοια φιλοσοφία και έγιναν γνωστές ως RISC (Masood). Κάποια από τα χαρακτηριστικά σχεδιασμού όπως ο χρόνος εκτέλεσης ενός κύκλου παρουσιάστηκαν να είναι κοινά σε όλους τους επεξεργαστές. Το γεγονός αυτό οφείλεται στη βελτιστοποίηση της κάθε εντολής στη CPU και σε μία τεχνική ονομάζεται Pipelining (Katevenis). Η τεχνική αυτή αφορά στην άμεση εκτέλεση μερών, ή καταστάσεων των εντολών έτσι ώστε να γίνει η βέλτιστη επεξεργασία τους.

Σημαντικός είναι επίσης και ο μεγάλος αριθμός των καταχωρητών πάνω στον οποίο βασίζεται η RISC φιλοσοφία με στόχο το να αποφύγει τις πολλές αλληλεπιδράσεις με τη μνήμη.

Στα πρώτα χρόνια της βιομηχανίας των υπολογιστών, ο προγραμματισμός γινόταν με γλώσσα assembly ή με γλώσσα μηχανής γεγονός που έκανε πολύ εύκολη τη χρήση των εντολών. Έτσι λοιπόν, οι σχεδιαστές της CPU προσπάθησαν να κάνουν εντολές τέτοιες που θα έκαναν όσο το δυνατό περισσότερη δουλειά. Η έλευση των γλωσσών υψηλότερου επιπέδου οδήγησε στη δημιουργία εντολών τέτοιων που θα ενεργοποιούν αυτόματα κάποιους μηχανισμούς των γλωσσών αυτών.

Ένας ακόμη στόχο ήταν η παροχή δυνατής διευθυνσιοδότησης για κάθε εντολή, γνωστή και ως ορθογωνιότητα η οποία διευκόλυνε την εφαρμογή του μεταγλωττιστή. Οι αριθμητικές λειτουργίες μπορούσαν λοιπόν να έχουν πλέον αποτελέσματα όπως και οι τελεστές άμεσα στη μνήμη (Katevenis). Η αντίληψη που επικρατούσε την περίοδο εκείνη ήταν ότι ο σχεδιασμός του υλικού ήταν πιο ώριμος από το σχεδιασμό του μεταγλωττιστή και αυτός ήταν ο λόγος για τον οποίο υλοποιούνταν μέρη της λειτουργικότητας και μικροκώδικας στο υλικό και όχι στη μνήμη του μεταγλωττιστή. Η φιλοσοφία αυτή του σχεδιασμού ονομάστηκε αυτόματα σύνθετη – complex, αμέσως μετά τη στιγμή που η φιλοσοφία RISC ήρθε στο προσκήνιο (Katevenis).




**ΕΙΚΟΝΑ 4 : ΤΥΠΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ CISC**

Ιδιαίτερα σημαντική σε ότι αφορά την αύξηση της πολυπλοκότητας ήταν η σημαντικά περιορισμένη μνήμη που έφτανε τα μόλις μερικά kilobytes. Για το λόγο αυτό η πυκνότητα της πληροφορίας που συγκρατούνταν στα υπολογιστικά προγράμματα ήταν υψηλή και οδηγούσε σε υψηλής κωδικοποίησης και μεταβλητού μεγέθους εντολές. Εξίσου σημαντικό ήταν και το γεγονός ότι η κύριες μνήμες ήταν σχετικά αργές. Οι σύγχρονοι υπολογιστές αντιμετωπίζουν και αυτοί παρόμοια προβλήματα καθώς οι κύριες μνήμες τους είναι αρκετά αργές σε σχέση με τη CPU ενώ οι γρήγορες cache μνήμες είναι περιορισμένες σε μέγεθος. Τα παραπάνω εξηγούν μερικώς για ποιο λόγο οι εντολές υψηλής κωδικοποίησης αποδείχθηκαν ιδιαίτερα χρήσιμες για τους σύγχρονους υπολογιστές ακριβώς όπως οι RISC σχεδιασμοί.

Οι σχεδιαστές των αρχιτεκτονικών κάνουν τις διάφορες επιλογές τους βασισμένοι στην ήδη υπάρχουσα τεχνολογία. Έτσι, όσο εξελίσσεται η τεχνολογία σε ότι αφορά στο υλικό και το λογισμικό εξελίσσονται και οι σχεδιασμοί. Επιπλέον, η εμπειρία που αποκτάται στο σχεδιασμό των επεξεργαστών οδηγεί και στο σχεδιασμό καλύτερων συστημάτων.

Η προσέγγιση της RISC είναι μία απόκριση στη συνεχώς εναλλασσόμενη τεχνολογία και στην συνάθροιση της γνώσης που προερχόταν από τους CISC σχεδιασμούς (Katevenis). Οι CISC επεξεργαστές σχεδιάζονταν για να απλοποιήσουν τους compilers και για να βελτιώσουν την απόδοση υπό τους διάφορους περιορισμούς όπως ήταν οι μικρές και αργές μνήμες. Οι σημαντικότεροι λόγοι που παρακίνησαν τους σχεδιαστές να μελετήσουν αρχιτεκτονικούς σχεδιασμούς διαφορετικούς της CISC ήταν (Masood):

- Απλούστερες εντολές : Οι σχεδιαστές των CISC αρχιτεκτονικών προέβαιναν σε εκτεταμένη χρήση των σύνθετων εντολών λόγω του ότι αυτές μείωναν το σημασιολογικό κενό. Στην πραγματικότητα φαίνεται ότι οι μεταγλωττιστές κυρίως αγνοούν τις εντολές αυτές και ένας λόγος για τον οποίο συμβαίνει αυτό είναι ότι οι διαφορετικές γλώσσες υψηλού επιπέδου χρησιμοποιούν και διαφορετικές σημασιολογίες (Masood).
- Λιγότεροι τύποι δεδομένων : Η CISC ISA υποστηρίζει μία πληθώρα δομών δεδομένων, από απλούς τύπους όπως είναι οι ακέραιοι και οι χαρακτήρες έως και σύνθετες δομές δεδομένων όπως τα αρχεία και οι δομές. Η εμπειρία υποστηρίζει ότι οι σύνθετες δομές δεδομένων δε χρησιμοποιούνται και τόσο συχνά έτσι λοιπόν κρίνεται απαραίτητο να σχεδιαστούν συστήματα που

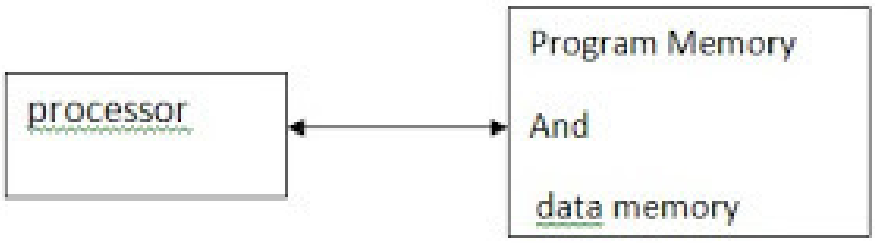
υποστηρίζουν απλούστερους τύπους δεδομένων αποτελεσματικά και έχουν τη δυνατότητα να συνθέτους τους δύσκολους τύπους δεδομένων που τους υπολείπονται (Masood).

- Απλούστεροι τρόποι διευθυνσιοδότησης : Οι σχεδιασμοί CISC παρέχουν έναν μεγάλο αριθμό τρόπων διευθυνσιοδότησης. Οι βασικοί λόγοι για αυτό είναι η υποστήριξη των σύνθετων δομών δεδομένων αλλά και η παροχή ευελιξίας στους τελεστές πρόσβασης (Masood).

### 2.2.2 ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ VON NEUMANN ΚΑΙ HARVARD

Υπάρχουν δύο βασικοί τύποι αρχιτεκτονικών σε ότι αφορά στους ψηφιακούς υπολογιστές. Η πρώτη ονομάζεται Von Neumann architecture ενώ η δεύτερη Harvard και υιοθετήθηκε αργότερα αποκλειστικά και μόνο για το σχεδιασμό ψηφιακών υπολογιστών.

- **Αρχιτεκτονική Von Neumann :**



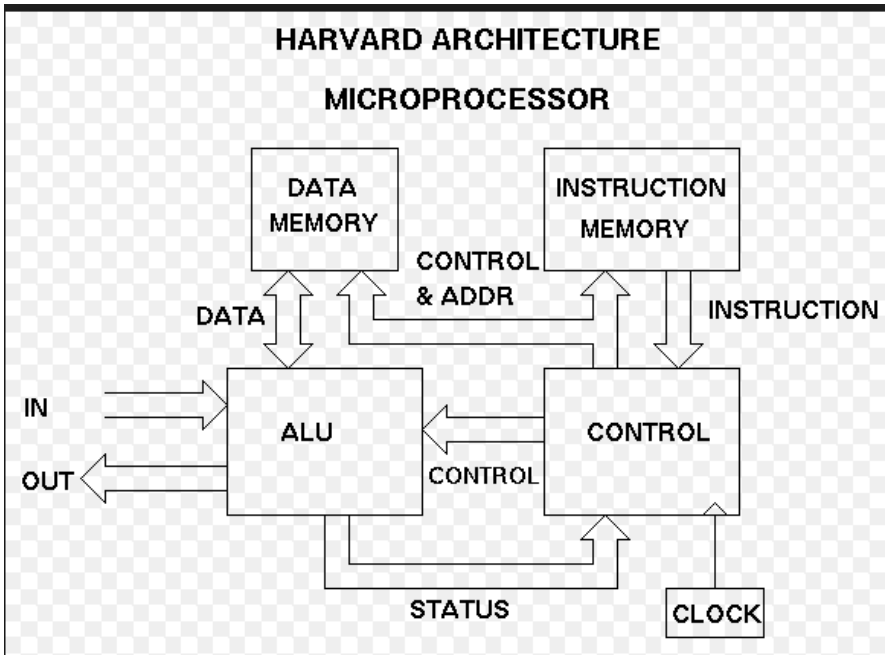

**ΕΙΚΟΝΑ 5 : ΑΡΧΙΤΕΚΤΟΝΙΚΗ VON NEUMANN**

Χαρακτηριστικά της αρχιτεκτονικής (Wikipedia):

- ✓ Πήρε το όνομά της από τον μαθηματικό και επιστήμονα των υπολογιστών John Von Neumann.
- ✓ Ο υπολογιστής έχει μία και μόνο μνήμη για να αποθηκεύει τα δεδομένα και το πρόγραμμα όταν αυτό εκτελείται.

- ✓ Ο επεξεργαστής χρειάζεται δύο κύκλους ρολογιού για να ολοκληρώσει μία εντολή. Το Pipelining των εντολών δεν είναι εφικτό σε αυτήν την αρχιτεκτονική.
- ✓ Κατά τον πρώτο κύκλο του ρολογιού ο επεξεργαστής λαμβάνει την εντολή από τη μνήμη και την αποκωδικοποιεί. Στον επόμενο κύκλο λαμβάνονται τα απαραίτητα από τη μνήμη δεδομένα. Ο κύκλος επαναλαμβάνεται για κάθε εντολή και έτσι απαιτούνται δύο κύκλοι για να ολοκληρωθεί μία εντολή.
- ✓ Πρόκειται για μία ιδιαίτερα παλιά αρχιτεκτονική που αντικαταστάθηκε από την αρχιτεκτονική Harvard.

- **Harvard Architecture:**




**ΕΙΚΟΝΑ 6 : ΑΡΧΙΤΕΚΤΟΝΙΚΗ HARVARD**

Χαρακτηριστικά της αρχιτεκτονικής (Wikipedia):

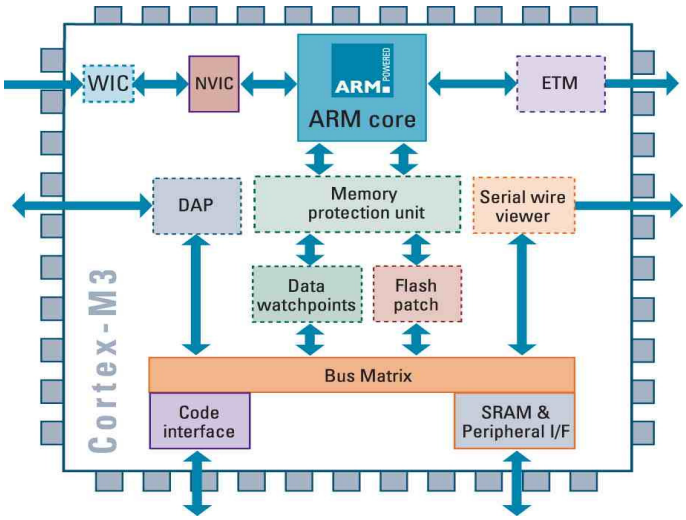
- Πήρε το όνομά της από τον "Harvard Mark I" έναν παλαιό υπολογιστή
- Ο επεξεργαστής μπορεί να εκτελέσει μία εντολή σε ένα κύκλο εάν εφαρμοστούν οι κατάλληλες pipelining στρατηγικές

- Στο πρώτο στάδιο του pipelining οι εντολές που εκτελούνται μπορούν να ληφθούν από τη μνήμη του προγράμματος. Στο δεύτερο στάδιο τα δεδομένα λαμβάνονται από τη μνήμη δεδομένων με χρήση των εντολών αποκωδικοποίησης
- Οι περισσότερες από τις μοντέρνες αρχιτεκτονικές υπολογιστών βασίζονται στην Harvard αλλά διαφέρει ο αριθμός των σταδίων του pipeline από σύστημα σε σύστημα

## 2.2.3 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ARM

### 2.2.3.1 ΓΕΝΙΚΑ

Η αρχιτεκτονική ARM εξελίχθηκε τα τελευταία χρόνια και έχει φθάσει σε σημείο τέτοιο ώστε να υποστηρίζει υλοποιήσεις ενός μεγάλου φάσματος σημείων απόδοσης. Η αρχιτεκτονική απλότητα των επεξεργαστών ARM οδήγησαν σε υλοποιήσεις πολύ μικρού μεγέθους, και υλοποιήσεις που επιτρέπουν τη χρήση συσκευών με πολύ χαμηλή κατανάλωση ενέργειας (ARM Architecture reference manual, 2005). Το μέγεθος, η απόδοση και η πολύ χαμηλή κατανάλωση ενέργειας αποτελούν τα βασικά χαρακτηριστικά εξέλιξης της αρχιτεκτονικής ARM.




**ΕΙΚΟΝΑ 7 : ΑΡΧΙΤΕΚΤΟΝΙΚΗ ARM**

Η ARM ανήκει στην κατηγορία RISC καθώς περιλαμβάνει κάποια από τα βασικά χαρακτηριστικά των αρχιτεκτονικών αυτών όπως (ARM White Paper, 2014):

- Ένα μεγάλο ενιαίο αρχείο καταγραφή
- Μία αρχιτεκτονική load/store όπου οι λειτουργίες επεξεργασίας των δεδομένων λειτουργούν σε πλαίσια καταχώρησης και όχι άμεσα επάνω στα πλαίσια των μνημών
- Απλοί τρόποι διευθυνσιοδότησης, με όλες τις load/store διευθύνσεις να καθορίζονται από τους καταχωρητές και τα πεδία των εντολών
- Ενιαία και σταθερού μεγέθους πεδία εντολών ώστε να απλοποιείται η αποκωδικοποίση

Επιπλέον η αρχιτεκτονική ARM παρέχει (ARM White Paper, 2014):

- Έλεγχο στην ALU (Arithmetic Logic Unit) και shifter στις περισσότερες εντολές επεξεργασίας δεδομένων έτσι ώστε να μεγιστοποιηθεί η χρήση σε μία ALU και έναν shifter
- Τρόπους διευθυνσιοδότησης αυτό – αύξησης και αυτό – μείωσης έτσι ώστε να βελτιστοποιηθούν οι βρόγχοι των προγραμμάτων
- Πολλαπλές εντολές Load και Store έτσι ώστε να βελτιστοποιηθεί το throughput των δεδομένων
- Εκτέλεση κατά συνθήκη όλων των εντολών έτσι ώστε να μεγιστοποιηθεί το throughput της εκτέλεσης

Οι βελτιώσεις αυτές σε μία βασική RISC επιτρέπουν στους επεξεργαστές ARM να επιτυγχάνουν ιδιαίτερα υψηλές αποδόσεις, μικρό μέγεθος κώδικα, χαμηλή κατανάλωση ενέργειας και μικρή περιοχή πυριτίου.

### 2.2.3.2 ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Κάθε ARM έχει 31 γενικού σκοπού καταχωρητές 32bit. Από αυτούς μονάχα οι 16 είναι ορατοί ενώ οι υπόλοιποι χρησιμοποιούνται για να επιταχύνουν τη διαδικασία των εξαιρέσεων. Έτσι λοιπόν, όλες οι εντολές απευθύνονται σε οποιονδήποτε από αυτούς τους 16 ορατούς καταχωρητές (ARM Architecture reference manual, 2005). Ο

κύριος αποθηκευτικός χώρος των 16 αυτών καταχωρητών χρησιμοποιήται από τον «φτωχό» κώδικα. Οι καταχωρητές αυτοί ανήκουν στο User Mode που είναι διαφορετικός από τους υπόλοιπους modes στα παρακάτω (ARM Architecture reference manual, 2005) :

- Το User Mode μπορεί να μεταβεί σε άλλο mode επεξεργασίας μονάχα δημιουργώντας ένα exception
- Τα συστήματα μνήμης και οι συνεπεξεργαστές δίνουν μικρότερη πρόσβαση στη μνήμη και μικρότερη λειτουργικότητα στους συνεπεξεργαστές

Η αρχιτεκτονική ARM υποστηρίζει επτά τύπους exception και ένα προηγμένο τρόπο εξεργασίας για τον κάθε ένα από αυτούς. Οι επτά τύποι είναι (ARM White Paper, 2014):

- Reset
- SWI (Software Interrupt Instructions)
- Prefetch abort
- Data abort
- IRQ
- FIQ

Κάθε φορά που προκύπτει ένα exception κάποιοι από τους καταχωρητές αντικαθίστανται με καταχωρητές ειδικούς να λειτουργούν σε exception mode. Ο επεξεργαστής με τη σειρά του σταματά την εκτέλεση με έναν καθορισμένο τρόπο και ξεκινά την εκτέλεση σε μία από τις σταθερές διευθύνσεις της μνήμης γνωστές και ως exception vectors. Υπάρχει ξεχωριστό vector για κάθε exception η συμπεριφορά του οποίου καθορίζεται από τα συστήματα εκτέλεσης (ARM Architecture reference manual, 2005).

Όλες οι καταστάσεις του επεξεργαστή πέρα από αυτές του γενικού περιεχομένου διατηρούνται στους καταχωρητές κατάστασης. Η τρέχουσα κατάσταση λειτουργίας του επεξεργαστή βρίσκεται στον CPSR (Current Program Status Register) ο οποίος διαθέτει (ARM Architecture reference manual, 2005) :

- Τέσσερα flag κατάστασης (Negative, Zero, Carry και oVerflow).
- Ένα sticky (Q) flag
- Τέσσερα GE flags (Greater than ή Equal)
- Δύο bit απενεργοποίησης διακοπών

- Πέντε bits τα οποία κωδικοποιούν την τρέχουσα κατάσταση του επεξεργαστή
- Δύο bits που κωδικοποιούν εάν εκτελούνται ARM, Thumb ή Jazelle εντολές
- Ένα bit το οποίο ελέγχει το φόρτο και τις διαδικασίες αποθήκευσης

Η ομάδα των εντολών που υποστηρίζεται από την ARM αρχιτεκτονική μπορεί να χωριστεί σε 6 διαφορετικές κλάσεις.

1. Βασικές εντολές
2. Εντολές επεξεργασίας δεδομένων
3. Εντολές μεταφοράς κατάστασης του καταχωρητή
4. Εντολές load και store
5. Εντολές συνεπεξεργαστή
6. Εντολές παραγωγής exception

Σχεδόν όλες οι κατηγορίες των ARM εντολών περιέχουν ένα πεδίο κατάστασης 4bit.

Ιδιαίτερα σημαντικό κομμάτι της αρχιτεκτονικής ARM αποτελεί η τεχνολογία Thumb. Η τεχνολογία αυτή αποτελείται από σετ οδηγιών 16 bit που λειτουργούν βοηθητικά για την ομάδα των 32 bit οδηγιών της κλασικής ARM (Embedded). Κάθε εντολή της Thumb μπορεί να εκτελεστεί διαμέσου μιας 32 bit εντολής ARM, οι οποίες όμως δεν είναι όλες διαθέσιμες για τις Thumb. Επιπλέον, κάποιες από τις εντολές που μπορούν να υλοποιηθούν σε μία ARM μπορούν μονάχα να εξομοιωθούν με μία ακολουθία από ARM εντολές (Embedded).

Στο σημείο αυτό θα μπορούσε κανείς να αναρωτηθεί για ποιο λόγο χρειάζονται δύο ομάδες εντολών στην ίδια CPU. Στην πραγματικότητα περιέχεται μονάχα ένα σετ εντολών και αυτό είναι των 32 bit. Όταν οι εντολές αυτές λειτουργούν στην κατάσταση Thumb, τότε ο επεξεργαστής απλά επιλέγει τις εντολές που καταλαμβάνουν μικρότερο χώρο στη μνήμη και είναι αντίστοιχες με αυτές των 32 bit (Embedded).

Η διαφορά ανάμεσα στις δύο εντολές έγκειται στο πως αυτές ερμηνεύονται πριν την εκτέλεση του προγράμματος και όχι στο πως αυτές λειτουργούν. Από τη στιγμή που η

επέκταση της εντολής 16 bit σε εντολή 32 bit γίνεται μέσα σε συγκεκριμένο υλικό του τσιπ, δεν επηρεάζει καθόλου την εκτέλεση και δεν την καθυστερεί σε κάνενα σημείο (Embedded).

Το σετ των εντολών Thumb παρέχει το μεγαλύτερο μέρος της λειτουργικότητας που απαιτείται για μία τυπική εφαρμογή. Υποστηρίζονται λειτουργίες όπως οι λογικές και οι αριθμητικές, οι εντολές φόρτωσης και αποθήκευσης των δεδομένων αλλά και πολλές άλλες συνθήκες. Ο κώδικας των εντολών αυτών μπορεί να γραφεί σε γλώσσα C και να εκτελεστεί άψογα σε κατάσταση Thumb.

Σε ότι αφορά τους καταχωρητές τώρα, θα πρέπει να σημειωθεί ότι, οι 12 από αυτούς στους οποίους έχει πρόσβαση η Thumb είναι ακριβώς οι ίδιοι που είναι προσβάσιμοι και από την τεχνολογία ARM και έτσι τα δεδομένα μπορούν να μεταφερθούν από ένα λογισμικό που χρησιμοποιεί ARM σε ένα λογισμικό που χρησιμοποιεί Thumb και αντίστροφα (Embedded).

Υπάρχουν πολλοί και διάφοροι τρόποι ώστε να γίνει κατάλληλα είσοδος ή έξοδος από την κατάσταση Thumb, η πιο συχνή όμως είναι η εντολή Branch and Exchange (BX). Κατά τη διάρκεια της εντολής αυτής, η CPU εξετάζει το bit που είναι λιγότερο σημαντικό από τα άλλα ώστε να καθορίσει τη διεύθυνση προορισμού της νέας κατάστασης. Έτσι λοιπόν εάν τιμή του bit είναι 1 τότε ο επεξεργαστής αλλάζε σε κατάσταση Thumb πριν αρχίσει να λειτουργεί στη νέα διεύθυνση ενώ εάν η τιμή του bit είναι 0 και βρίσκεται στην κατάσταση Thumb επιστρέφει σε κατάσταση ARM (Embedded).

Ο κυριότερος λόγος για τον οποίο κάποιος θα αναζητήσει έναν επεξεργαστή ARM που θα περιλαμβάνει και το σετ οδηγιών Thumb θα είναι η μείωση στην πυκνότητα

του κώδικα. Επιπλέον, μειώνεται ο συνολικός χώρος που απαιτείται στη μνήμη καθώς ο δίαυλος δεδομένων μπορεί να μειωθεί στα 16 bit.

Η ομάδα εντολών Thumb2 αποτελεί μία βελτίωση της ήδη υπάρχουσας 16 bit Thumb και επιτρέπει την ανάμειξη των εντολών 32 bit με αυτές των 16 bit σε ένα και μόνο πρόγραμμα. Οι επιπλέον εντολές των 32 bit οι οποίες χρησιμοποιούνται από την Thumb2, καλύπτουν όλη την απαιτούμενη από την ARM λειτουργικότητα (ARM Infocenter). Έτσι υπάρχει η δυνατότητα συνδυασμού κώδικα και με παλαιότερες εκδόσεις όταν την ίδια στιγμή παρέχεται απόδοση αντίστοιχη των εντολών της ARM.

Η σημαντικότερη διαφορά ανάμεσα στις ομάδες εντολών Thumb2 και ARM βρίσκεται στο γεγονός ότι οι περισσότερες από τις 32 bit εντολές Thumb λειτουργούν χωρίς όρους και περιορισμούς ενώ οι αντίστοιχες της ARM λειτουργούν υπό όρους (ARM Infocenter).

Οι εντολές της Thumb2 ακριβώς όπως και της Thumb είναι προσβάσιμες όταν ο επεξεργαστής βρίσκεται σε κατάσταση Thumb, όταν δηλαδή το λιγότερο σημαντικό bit βρίσκεται σε κατάσταση 1.

Οι βασικές βελτιώσεις που έχουν γίνει στην Thumb και περιέχονται στην Thumb2 είναι (ARM Infocenter) :

- Πρόσθεση εντολών 32 bit που ώστε να :
    - Γίνει καλύτερος χειρισμός των εξαιρέσεων
    - Προσφερθεί πρόσβαση στους επεξεργαστές
    - Συμπεριληφθεί ψηφιακή επεξεργασία σήματος
    - Βελτιωθεί η απόδοση σε περιπτώσεις όπου υπάρχουν περιορισμοί από τις εντολές 16bit

- Προσθήκη εντολών 16bit που βελτιώνουν κατά πολύ τις ήδη υπάρχουσες Thumb και τις καθιστούν να λειτουργούν υπό όρους
- Προσθήκη μιας εντολής 16bit Compare with Zero and Branch ώστε να βελτιωθεί η πυκνότητα του κώδικα

Η διαθεσιμότητα τόσο των εντολών 16 bit όσο και των εντολών 32 bit δίνει τη δυνατότητα της έμφασης τόσο στην απόδοση όσο και στο μέγεθος του κώδικα σύμφωνα πάντα με τις απαιτήσεις της εφαρμογής.

### 2.2.3.3 CORTEX M 0+

Ο επεξεργαστής ARM® Cortex®-M0+ είναι ο πλέον ενεργειακά αποδοτικός επεξεργαστής που είναι αυτή τη στιγμή διαθέσιμος. Έχει κατασκευαστεί με πρότυπο τον ήδη πολύ επιτυχημένο Cortex – M0 επεξεργαστή και διατηρεί την πλήρη συμβατότητα σε ότι αφορά τα εργαλεία και τις εντολές ενώ καταφέρνει να μειώσει ακόμη περισσότερο την κατανάλωση σε ενέργεια και αυξάνει την απόδοση. Όπως και στον Cortex – M0 η πολύ μικρή περιοχή πυριτίου, η χαμηλή κατανάλωση ισχύος και το ελάχιστο αποτύπωμα κώδικα αυτών των επεξεργαστών επιτρέπει στους προγραμματιστές να επιτύχουν μία απόδοση 32bit σε επεξεργαστές τιμών αναλόγων αυτών των 8bit (ARM). Ο επεξεργαστής Cortex M0+ περιλαμβάνει ένα πολύ μεγάλο αριθμό εφαρμογών ώστε να προσφέρει ιδιαίτερα ευέλικτη ανάπτυξη εφαρμογών. Η ομάδα των εντολών που χρησιμοποιείται είναι πλήρης και συμβατή με τις προηγούμενες εκδόσεις ώστε να χρησιμοποιούνται τα ίδια εργαλεία σε ότι αφορά το compile και το debug. Το pipeline του Cortex M0+ έχει μειωθεί από 3 σε 2 στάδια τα οποία μειώνουν και την κατανάλωση ισχύος (ARM). Στον Cortex M0+ έχουν επίσης προστεθεί τα χαρακτηριστικά των Cortex-M3 και Cortex-M4.

Τα βασικά χαρακτηριστικά του Cortex M0+ είναι (ARM):

- Αρχιτεκτονική ARMv6-M
- Pipeline 2 σταδίων

- Σετ εντολών :
  - Thumb, χωρίς τις CBZ, CBNZ, IT.
  - Thumb-2, μόνο οι BL, DMB, DSB, ISB, MRS, MSR.
  - 32-bit hardware πολλαπλασιαζόμενο με 32-bit αποτέλεσμα
- 1 έως 32 διακοπές

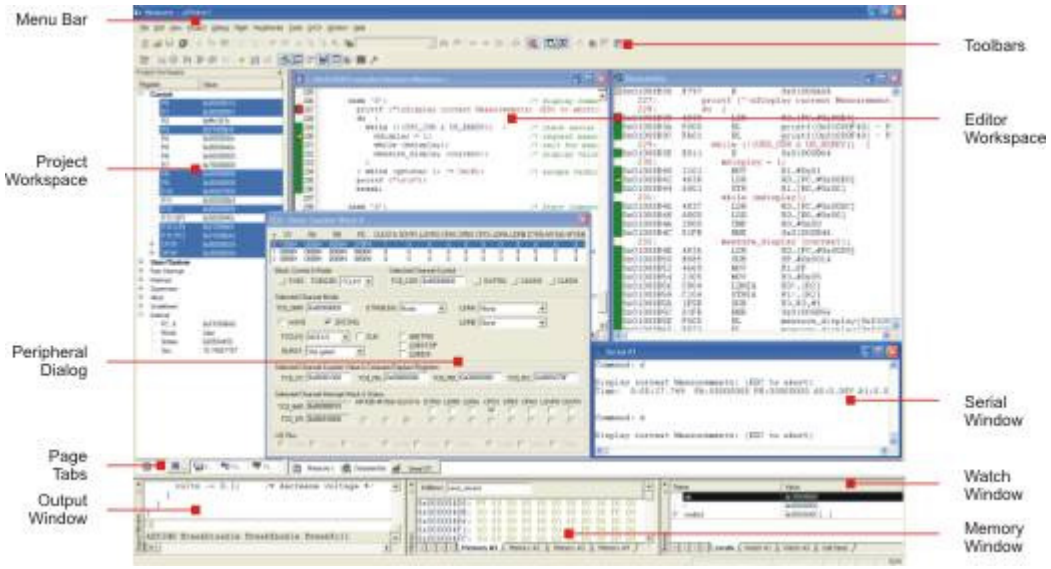
Οι παρακάτω μικροελεγκτές βασίζονται στον Cortex-M0+ (ARM):

- Atmel SAMD, SAMR, SAML, SAMC
- Freescale Kinetis E, EA, L, M, V1, W0
- NXP LPC800, LPC11E6x, LPC11U6x
- Silicon Labs/Energy Micro EFM32 Zero, Happy
- Spansion FM0+
- STMicroelectronics STM32 L0

# ΚΕΦΑΛΑΙΟ 3$^{Ο}$ - ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΕΝΣΩΜΑΤΩΜΕΝΩΝ

## 3.1 μVision της Keil

Το μVision αποτελεί μία πλατφόρμα ανάπτυξης λογισμικού που βασίζεται σε παράθυρα και συνδυάζει έναν ισχυρό και μοντέρνο editor με έναν project manager. Ενσωματώνει όλα εκείνα τα απαραίτητα εργαλεία για να αναπτυχθούν ενσωματωμένες εφαρμογές συμπεριλαμβανομένου ενός C/C++ compiler, ενός macro assembler, ενός linker/locator και μιας μηχανής παραγωγής HEX αρχείων (Keil).




**ΕΙΚΟΝΑ 8 : ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΤΟΥ ΜVISION**

Το μVision επισπεύδει σημαντικά τη διαδικασία ανάπτυξης των ενσωματωμένων εφαρμογών παρέχοντας τα ακόλουθα (Getting Started With Keil) :

- Code editor με άπειρες δυνατότητες και χαρακτηριστικά
- Βάση δεδομένων διαφόρων συσκευών που βοηθά στη διαμόρφωση του εργαλείου ανάπτυξης
- Project manager για τη δημιουργία και τη συντήρηση των projects
- Ενσωματωμένη λειτουργία Make Utility για τη συναρμολόγηση, το compile και τη σύνδεση των ενσωματωμένων εφαρμογών
- Παράθυρα διαλόγων για όλες τις ρυθμίσεις του περιβάλλοντος ανάπτυξης
- Ενσωματωμένο Debugger με CPU υψηλής ταχύτητας και Simulator
- Προηγμένη GDI διεπαφή για software debugging στο υλικό – στόχο και για τη σύνδεση με έναν Keil™ ULINK®Debug Adapter

- Λειτουργία Flash programming για κατέβασμα του προγράμματος της εφαρμογής σε μία Flash Rom
- Συνδέσμους με τα εγχειρίδια χρήσης, την on line βοήθεια και τους οδηγούς χρήσης

Το IDE αλλά και ο Debugger αποτελούν τα βασικά μέρη των εργαλείων ανάπτυξης της Keil και έχουν έναν μεγάλο αριθμό χαρακτηριστικών που βοηθούν των προγραμματιστή να αναπτύξει ενσωματωμένες εφαρμογές γρήγορα και επιτυχημένα.

Τα εργαλεία είναι ιδιαίτερα εύκολα στο να χρησιμοποιηθούν και βοηθούν σημαντικά στην επίτευξη αποτελεσματικών σχηματισμών σε πολύ μικρό χρονικό διάστημα (Getting Started With Keil).

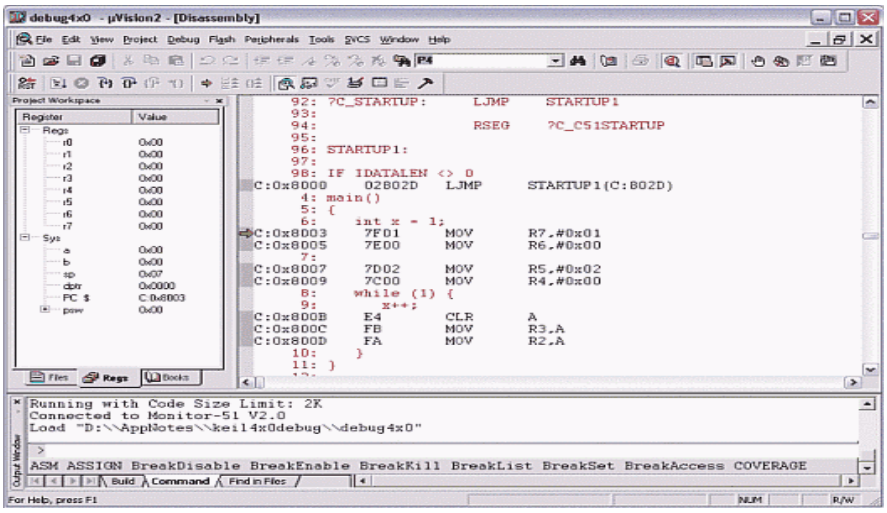
Το μVision ανάμεσα στα άλλα προσφέρει και ένα Build Mode για τη δημιουργία εφαρμογών και ένα Debug Mode για το debugging των εφαρμογών αυτών. Στις εφαρμογές μπορεί επίσης να γίνει debug και με τον ενσωματωμένο μVision προσομοιωτή. Οι προγραμματιστές μπορούν επίσης να χρησιμοποιήσουν και τους AGDI Adapters ή εξωτερικά εργαλεία για να κάνουν ανάλυση των εφαρμογών τους (Getting Started With Keil).

Το μενού Device Database προσφέρει έναν ιδιαίτερα εύχρηστο τρόπο με τον οποίο μπορεί ο χρήστης να διαμορφώσεις τις διάφορες συσκευές αλλά και τις παραμέτρους του project. Επιπλέον, μπορεί να γίνει χρήση συσκευών που θα προστεθούν από τον ίδιο το χρήστη ή να αλλαχθούν οι τρέχουσες διαμορφώσεις.

Ο Debugger είναι απόλυτα ενσωματωμένος στο IDE και έχει τα παρακάτω χαρακτηριστικά (Getting Started With Keil):

- ✓ Αποσύνθεση του κώδικα C/C++ ή σύνθεση αυτού με ταυτόχρονη εκτέλεση του προγράμματος σε διάφορα stepping modes
- ✓ Πολλαπλές επιλογές breakpoint συμπεριλαμβανομένων των σύνθετων και της πρόσβασης
- ✓ Βοηθητικά bookmarks για την ανεύρεση των κρίσιμων σημείων
- ✓ Αναθεώρηση και διαμόρφωση της μνήμης, των μεταβλητών και των τιμών του καταχωρητή
- ✓ Λίστα του δέντρου του προγράμματος που περιλαμβάνει τις μεταβλητές της στοίβας
- ✓ Αναθεώρηση της κατάστασης των on chip περιφερειακών του μικροελεγκτή

- ✓ Εντολές debugging
- ✓ Προφίλ εκτέλεσης ώστε να καταγράφεται και να παρουσιάζεται ο χρόνος που καταναλώνεται αλλά και οι κύκλοι που χρειάζονται για κάθε εκτέλεση
- ✓ Στατιστικά κάλυψης του κώδικα για έλεγχο των σημαντικών εφαρμογών
- ✓ Διάφορα εργαλεία ανάλυσης
- ✓ Εργαλεία παρουσίασης ιστορικού της εκτέλεσης των εντολών




**ΕΙΚΟΝΑ 9 : Ο ΜVISION DEBUGGER**

Το πρόγραμμα διαθέτει έναν αναλυτή που επιτρέπει τη συγγραφή προγραμμάτων με χρήση των εντολών του μικροελεγκτή. Χρησιμοποιείται σε εφαρμογές όπου η ταχύτητα, το μικρό μέγεθος κώδικα αλλά και ο ακριβής έλεγχος του υλικού είναι απαραίτητα. Επιπλέον παρέχεται η δυνατότητα μάκρο-επεξεργασίας με πολύ ισχυρές δυνατότητες. Οι αναλυτές της Keil υποστηρίζουν διαφορετικούς τύπους μάκρο – επεξεργαστών όπως ο Standard μάκρο – επεξεργαστής ή η γλώσσα μάκρο – επεξεργασίας (Keil).

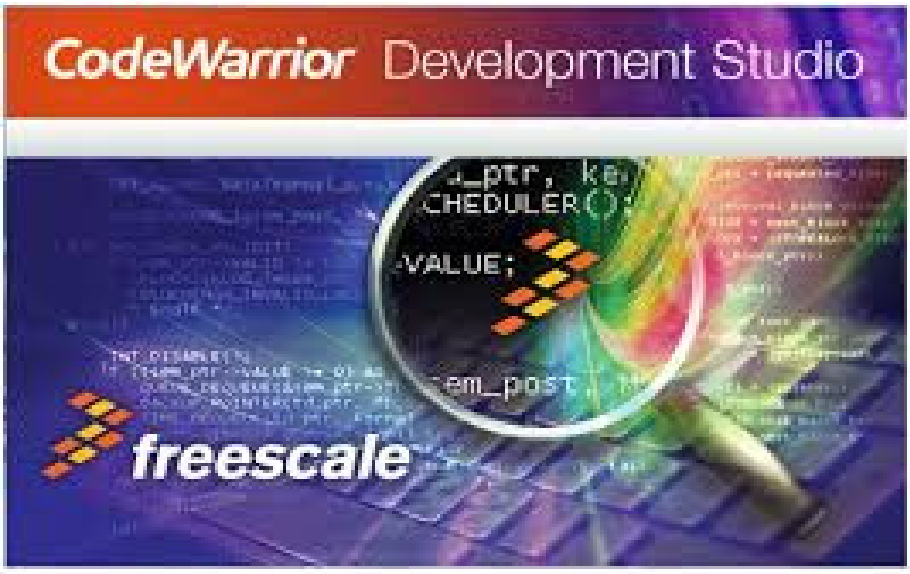
The ARM C/C++ compiler έχει σχεδιαστεί ώστε να παράγει γρήγορο και συμπαγή κώδικα για τις ARM7, ARM9 και Cortex-Mx αρχιτεκτονικές επεξεργαστών ενώ ο Keil ANSI C compiler στοχεύουν στις 8051, C166, XE166, and XC2000 αρχιτεκτονικές (Getting Started With Keil). Μπορούν να παράξουν αντικειμενοστραφή κώδικα που ταιριάζει με την αποτελεσματικότητα και την ταχύτητα των γλωσσών χαμηλού επιπέδου. Η χρήση μιας γλώσσας υψηλού επιπέδου όπως η C/C++ προσφέρει πολλά πλεονεκτήματα όπως (Getting Started With Keil) :

- Δεν απαιτείται γνώση των εντολών του επεξεργαστή
- Η στοιχειώδης γνώση της αρχιτεκτονικής του μικροελεγκτή είναι επιθυμητή αλλά δεν είναι απαραίτητη
- Λεπτομέρειες όπως η δέσμευση του καταχωρητή, η διευθυνσιοδότηση διαφόρων τύπων μνήμης, και διευθυνσιοδότηση διαφόρων τύπων δεδομένων διαχειρίζονται από τον compiler
- Τα προγράμματα έχουν μία τυπική δομή και μπορούν να διαχωριστούν σε ξεχωριστές συναρτήσεις
- Μπορούν να χρησιμοποιηθούν λέξεις κλειδιά και συναρτήσεις που βοηθούν την ανθρώπινη σκέψη
- Ο χρόνος ανάπτυξης του λογισμικού και του debugging μειώνονται σημαντικά

Ο μετατροπέας αντικειμένων σε δεκαεξαδικό παράγει Intel HEX αρχεία από object modules που έχουν δημιουργηθεί από τον linker. Τα αρχεία Intel HEX είναι ASCII αρχεία που περιλαμβάνουν δεκαεξαδική απεικόνιση της εφαρμογής. Φορτώνονται πολύ εύκολα στο πρόγραμμα της συσκευής ώστε να εγγραφούν στη ROM, EPROM, FLASH, ή οποιαδήποτε άλλη προγραμματιζόμενη μνήμη.

## 3.2 CodeWarrior της Freescale

Το CodeWarrior είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης που διανέμεται από την εταιρία Freescale με στόχο την ανάπτυξη, το compile και το debugging λογισμικού για διάφορους μικροελεγκτές και μικροεπεξεργαστές αλλά και ψηφιακούς ελεγκτές σήματος που χρησιμοποιούνται στα ενσωματωμένα συστήματα. Οι γλώσσες που υποστηρίζονται από το περιβάλλον αυτό είναι η C, η C++ και η Assembly (Wikipedia).

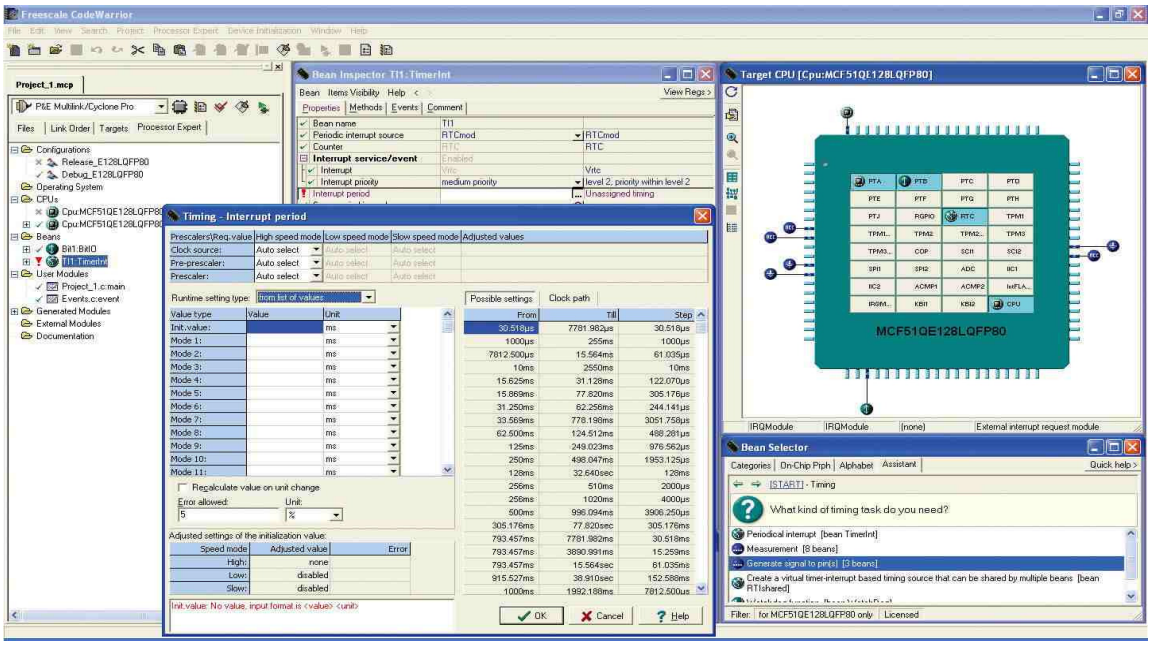



**ΕΙΚΟΝΑ 10 : CODEWARRIOR ΤΗΣ FREESCALE**

Πριν αναλάβει η Freescale τη διανομή του προϊόντος υπήρχαν πολλές και διαφορετικές εκδόσεις που στόχευαν σε Macintosh, Microsoft Windows, Linux, Solaris, PlayStation2, Nintendo GameCube, Nintendo DS, Wii, Sega Dreamcast, SuperH, M·CORE, Palm OS, Symbian OS, και BeOS. Οι εκδόσεις αυτές περιείχαν επίσης Pascal, Object Pascal, Objective-C, και Java compilers (Wikipedia).

Το CodeWarrior σχεδιάστηκε αρχικά από την εταιρία Metrowerks και βασιζόταν σε έναν C compiler και ένα περιβάλλον για τον Motorola 68K που είχε δημιουργηθεί από τον Andreas Hommel και είχε εξαγοραστεί από την Metrowerks. Οι πρώτες εκδόσεις του λογισμικού στόχευαν κυρίως στον PowerPC Macintosh με το μεγαλύτερο μέρος της ανάπτυξης του λογισμικού να έχει γίνει από την ομάδα THINK C (Wikipedia). Μάλιστα, το CodeWarrior ήταν πολύ γρηγορότερο από όλα τα λογισμικά ανάπτυξης που είχαν γραφεί από την Apple.

Αποτέλεσε σημαντικό παράγοντα επιτυχίας για τη μετάβαση της Apple από τους επεξεργαστές 68K στο PowerPC λόγω του ότι προσέφερε ένα σταθερό PowerPC Compiler τη στιγμή που ο ανταγωνισμός ήταν κατά βάση ημιτελής (Wikipedia). Από τη στιγμή που η Metrowerks εξαγοράστηκε από τη Motorola, το 1999, η εταιρία επικεντρώθηκε στις ενσωματωμένες εφαρμογές αφιερώνοντας ένα κομμάτι των προσπαθειών της σε compilers για σταθερούς υπολογιστές. Τον Ιούλιο του 2005, ανακοίνωσαν την έκδοση του CodeWarrior για Mac. Δυστυχώς η ζήτηση για το CodeWarrior έπεσε μερικώς από τη στιγμή που η Apple ξεκίνησε τη διανομή του XCode δωρεάν (Wikipedia).




**ΕΙΚΟΝΑ 11 : CODEWARRIOR ΤΗΣ FREESCALE**

Το όνομα του λογισμικού προέρχεται από την ταινία Mad Max 2 : The Road Warrior και δόθηκε στο λογισμικό από τον Greg Galanos. Το γρήγορο και ιδιαίτερα εύχρηστο αυτό λογισμικό περιλαμβάνει τα παρακάτω εργαλεία (Introduction to CodeWarrior) :

- Project Manager : Διαχειρίζεται τα αρχεία των τελευταίων επιπέδων, εντοπίζει την πληροφορία, καθορίζει τη σειρά με την οποία κατασκευάζονται και συμπεριλαμβάνονται τα αρχεία, συντονίζει τα plug-ins για να παρέχει υπηρεσίες όπως ο έλεγχος εκδόσεων

- Text Editor : Διευκολύνει τη δημιουργία και την επεξεργασία του πηγαίου κώδικα αλλά και άλλων αρχείων κειμένου
- Μηχανή αναζήτησης : Εντοπίζει ένα συγκεκριμένο string και το αντικαθιστά, επιτρέπει τη χρήση κανονικών εκφράσεων, δίνει τη δυνατότητα σύγκρισης αρχείων και διαφοροποίησης της λειτουργικότητας
- Source Browser : Διατηρεί μία βάση δεδομένων προτύπων για το πρόγραμμα, χρησιμοποιεί τη βάση αυτή δεδομένων για να βοηθήσει την πλοήγηση στον κώδικα, ενώνει κάθε σύμβολο με άλλες τοποθεσίες στον κώδικα που είναι σχετικές με αυτό, επεξεργάζεται τις αντικειμενοστραφείς και τις διαδικαστικές γλώσσες
- Build System : Χρησιμοποιεί τον compiler για να παράγει επανεντοπίσιμα αντικείμενα κώδικα από τον πηγαίο κώδικα και τον linker για να παράγει τελικές εκτελέσιμες εικόνες από τον αντικειμενοστραφή κώδικα. Το CodeWarrior Integrated Development Environment (IDE) χρησιμοποιεί projects για να οργανώσει τα αρχεία και διάφορες ρυθμίσεις για να δημιουργήσει ένα πρόγραμμα. Κάθε project είναι ένα αρχείο που χρησιμοποιεί έναν ή περισσότερους κατασκευαστικούς στόχους. Κάθε κατασκευαστικός στόχος σε ένα πρόγραμμα είναι μία συλλογή αρχείων του το IDE χρησιμοποιεί ώστε να φτιάξει ένα αρχείο εξόδου. Κάθε κατασκευαστικός στόχος μπορεί να περιέχει τα παρακάτω στοιχεία (Introduction to CodeWarrior):

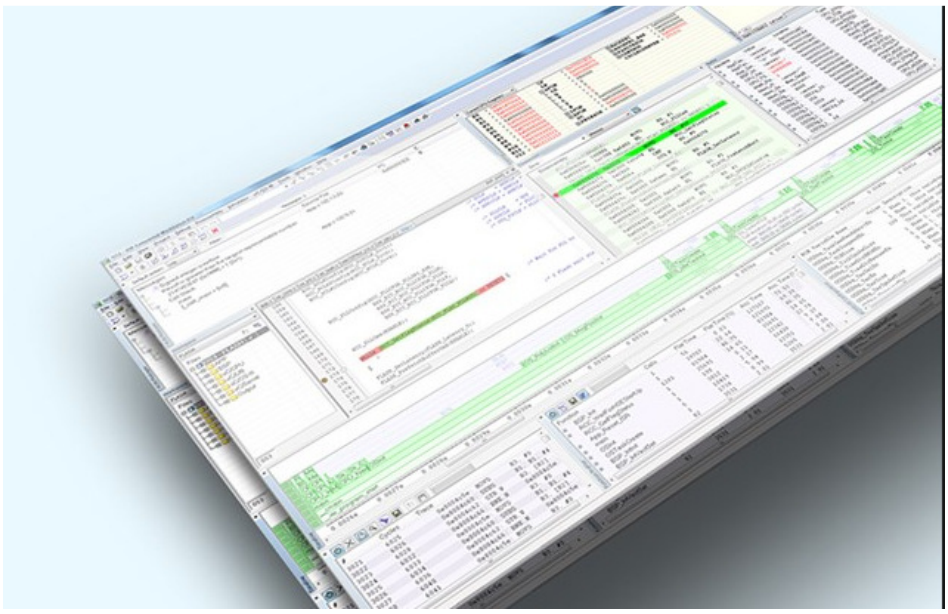
    - Αρχεία πηγαίου κώδικα
    - Βιβλιοθήκες
    - Ρυθμίσεις
    - Άλλα Projects που έχουν τα ίδια αρχεία

## 3.3 eWarm της IAR

Το ενσωματωμένο Workbench της IAR είναι ένα ιδιαίτερα ισχυρό ενσωματωμένο περιβάλλον ανάπτυξης (IDE) που επιτρέπει την ανάπτυξη και τη διαχείριση project ενσωματωμένων εφαρμογών (IAR C/C++ Development , 2009). Το περιβάλλον

ανάπτυξης είναι ιδιαίτερα φιλικό προς το χρήστη και εύκολο στη μάθηση ενώ την ίδια στιγμή παρέχει άπειρες δυνατότητες κληρονομικότητας κώδικα και υποστήριξη σε συγκεκριμένους στόχους.

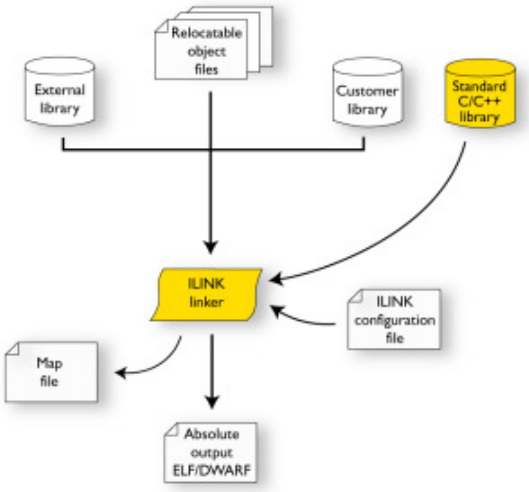
Το ενσωματωμένο Workbench της IAR προάγει μία πολύ χρήσιμη μεθοδολογία εργασίας και οδηγεί σε σημαντική μείωση του χρόνου ανάπτυξης. Ο compiler, ο debugger, ο linker και ο assembler μπορούν να εκτελεστούν και από ένα command line περιβάλλον εάν ο χρήστης επιθυμεί να τις χρησιμοποιήσει ως εξωτερικά εργαλεία (IAR C/C++ Development , 2009).



**ΕΙΚΟΝΑ 12 : EWARM**

Ο C/C++ Compiler της Arm προσφέρει τα βασικά χαρακτηριστικά τόσο της C όσο και της C++ μαζί με όποιες επεκτάσεις έχουν σχεδιαστεί για να μπορούν να εκτελούνται συγκεκριμένες ARM λειτουργίες (IAR C/C++ Development , 2009). Ο assembler της ARM είναι ένας ισχυρός macro assembler επανεντοπισμού με ένα ιδιαίτερα ευέλικτο σετ εντολών και τελεστών έκφρασης. Ουσιαστικά αποτελεί έναν χαρακτηριστικό προεπεξεργαστή της C και υποστηρίζει ανάλυση κατά συνθήκη. Χρησιμοποιεί την ίδια σύνταξη με τον Advanced RISC Machines Ltd ARM Assembler, ο οποίος απλοποιεί την μετακίνηση του υπάρχοντα κώδικα (IAR C/C++ Development , 2009).

Ο Linker αποτελεί και αυτός ένα ιδιαίτερα ισχυρό και ευέλικτο εργαλείο για χρήση στην ανάπτυξη εφαρμογών ενσωματωμένων ελεγκτών. Είναι κατάλληλος για την ένωση μικρών, μεμονωμένων αρχείων προγραμμάτων αλλά και για το linking μεγάλων input από C/C++ ή από συνδυασμό προγραμμάτων (IAR).




**ΕΙΚΟΝΑ 13 : EWARM LINKER**

Λόγω του ότι το ILINK χρησιμοποιεί και παράγει ELF και DWARF με τη μορφή αντικειμένων χρησιμοποιούνται κάποιες επιπλέον λειτουργίες που διαχειρίζονται αυτά τα format όπως (IAR):

● Το IAR Archive Tool—iarchive : Δημιουργεί και επεξεργάζεται μία βιβλιοθήκη διαφόρων ELF αρχείων αντικειμένων

● The IAR ELF Tool—ielftool : Επιτελεί διάφορους μετασχηματισμούς στα ELF εκτελέσιμα

● The IAR ARM ELF Dumper—ielfdumparm : Δημιουργεί μία αναπαράσταση κειμένου των περιεχομένων ενός ELF επανατοποθετήσιμου ή εκτελέσιμου

● The IAR ELF Object Tool—iobjmanip : Χρησιμοποιείται για την επεξεργασία χαμηλού επιπέδου σε ELF αρχεία αντικειμένων

● The IAR Absolute Symbol Exporter—isymexport : Εξάγει σύμβολα από ένα ROM αρχείο εικόνας έτσι ώστε αυτά να μπορούν να χρησιμοποιηθούν κατά το linking μίας add on εφαρμογής

Υπάρχουν δύο γλώσσες υψηλού επιπέδου οι οποίες μπορούν να χρησιμοποιηθούν στον IAR C/C++ compiler για ARM. Η πρώτη είναι η C, η οποία είναι και η πλέον χρησιμοποιημένη προγραμματιστική γλώσσα υψηλού επιπέδου στη βιομηχανία των κατανεμημένων συστημάτων (IAR). Η χρήση του compiler της IAR οδηγεί στη δημιουργία εφαρμογών για το πρότυπο ISO 9899:1990. Το πρότυπο αυτό είναι γνωστό και ως ANSI C. Η δεύτερη γλώσσα είναι η C++, μία αντικειμενοστραφής

γλώσσα προγραμματισμού με μία πλήρη βιβλιοθήκη κατάλληλη για τον δομημένο προγραμματισμό. Τα συστήματα της IAR υποστηρίζουν δύο επίπεδα της C++ (IAR).

Την κατανεμημένη C++, που αποτελεί υποσύνολο του υπολογιστικού προτύπου της C++ και χρησιμοποιείται για τον προγραμματισμό των ενσωματωμένων συστημάτων και την εκτεταμένη ενσωματωμένη C++ η οποία περιέχει έναν αριθμό επιπλέον χαρακτηριστικών όπως η πολλαπλή χωρητικότητα, οι τελεστές αλλά και η Standard Template Library (IAR). Κάθε μία από τις γλώσσες που υποστηρίζονται μπορεί να χρησιμοποιηθεί με αυστηρό ή και πιο χαλαρό τρόπο. Ακόμη, δίνεται η δυνατότητα υλοποίησης μέρους της εφαρμογής ή και ολόκληρης της εφαρμογής σε γλώσσα assembler.

Ο Compiler υποστηρίζει πολλές και διαφορετικές συσκευές που βασίζονται στις εκδόσεις 4, 5, 6, 6M, και 7. Ο κώδικας που παράγεται δεν είναι πάντα binary και συμβατός ανάμεσα στους cores. Έτσι είναι ιδιαίτερα σημαντικό να γίνει η κατάλληλη επιλογή επεξεργαστή στον compiler (IAR).

Η default εγκατάσταση του προγράμματος περιλαμβάνει έναν αριθμό έτοιμων αρχείων για υποστήριξη διαφόρων συσκευών. Τα header files για το I/O των περιφερειακών συσκευών έχουν την επέκταση h. Το πρόγραμμα παρέχει I/O αρχεία για όλες τις συσκευές που είναι διαθέσιμες έως και τη στιγμή της έκδοσής του. Τα αρχεία αυτά βρίσκονται στο φάκελο arm\inc\ (IAR).

Ο Debugger διαχειρίζεται διάφορες από τις απαιτήσεις που αφορούν στις συσκευές χρησιμοποιώντας αρχεία περιγραφής των συσκευών. Τα αρχεία αυτά βρίσκονται στο φάκελο arm\inc και έχουν την επέκταση ddf.

Ο φάκελος arm\examples περιλαμβάνει μερικές εκατοντάδες παραδειγμάτων έτσι ώστε να δώσουν στο χρήστη μία πρώτη ιδέα του προγράμματος καθώς και της ανάπτυξης των εφαρμογών. Η πολυπλοκότητα των παραδειγμάτων ποικίλει ξεκινώντας από τη χρήση ενός απλού LED έως και τους ελεγκτές μαζικής αποθήκευσης USB (IAR C/C++ Development , 2009). Τα παραδείγματα υποστηρίζονται από ένα πολύ μεγάλο αριθμό συσκευών ενώ ιδιαίτερη έμφαση δίνεται στα ενσωματωμένα συστήματα.

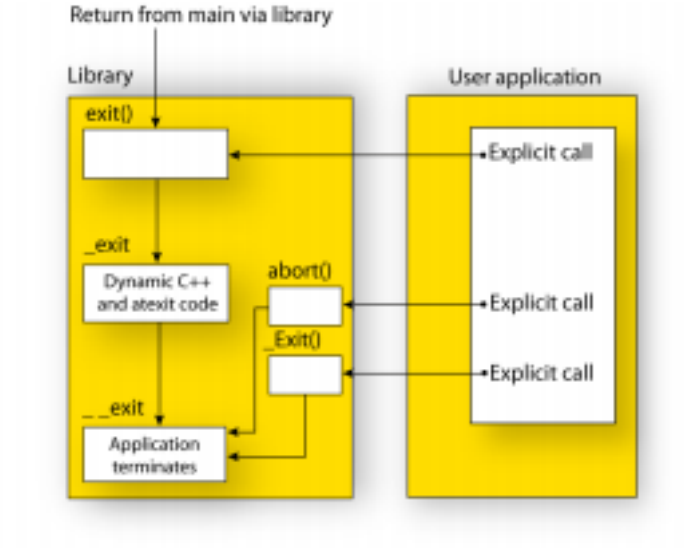
Ο compiler προσφέρει μία ομάδα από λέξεις – κλειδιά που μπορούν να χρησιμοποιηθούν για την παραγωγή του κώδικα. Για παράδειγμα υπάρχουν λέξεις

που καταδεικνύουν ειδικούς τύπου συναρτήσεων. Η επιλογή –e στο command line κάνει δυνατή την προβολή των εκτεταμένων λέξεων έτσι ώστε αυτές να μη μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών (IAR C/C++ Development , 2009).

Οι pragma οδηγίες ελέγχουν τη συμπεριφορά του compiler όπως για παράδειγμα το πώς αυτός δεσμεύει τη μνήμη, εάν επιτρέπει τις εκτεταμένες λέξεις και εάν παρουσιάζει μηνύματα προειδοποίησης. Οι οδηγίες pragma ενεργοποιούνται πάντα στον compiler, είναι συμβατές με ISO/ANSI C και είναι ιδιαίτερα χρήσιμες όταν θέλει κανείς να βεβαιωθεί ότι ο πηγαίος κώδικας μπορεί να μεταφερθεί.

Τα προκαθορισμένα σύμβολα του προεπεξεργαστή βοηθούν στην κατανόηση του περιβάλλοντος και του χρόνου του compile. Την ίδια στιγμή υπάρχουν συγκεκριμένα χαρακτηριστικά του hardware που υποστηρίζονται από τους ειδικούς τύπους συναρτήσεων του compiler, όπως οι διακοπές λογισμικού, οι διακοπές και οι γρήγορες διακοπές. Ο χρήστης μπορεί να γράψει μία πλήρη εφαρμογή χωρίς να χρειάζεται να γράψει καμία από τις παραπάνω συναρτήσεις στον assembler.

Για τα μέρη της εφαρμογής που σχετίζονται με το hardware απαιτείται πρόσβαση στα low level χαρακτηριστικά. Ο compiler υποστηρίζει πολλούς και διαφορετικούς τρόπους για να γίνει αυτό όπως οι εγγενείς συναρτήσεις, η ανάμειξη της C και των assembler modules αλλά και ο inline assembler (IAR).




**ΕΙΚΟΝΑ 14 : ΦΑΣΗ ΤΕΡΜΑΤΙΣΜΟΥ ΣΥΣΤΗΜΑΤΟΣ**

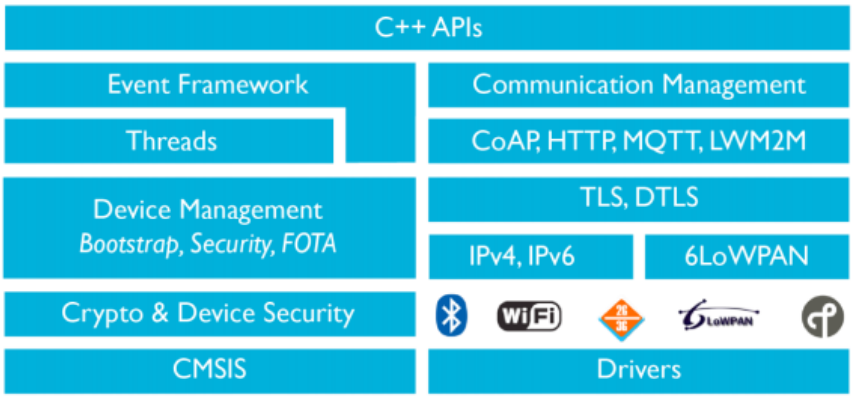
## 3.4 online compiler ARM MBED

Το Internet of Things αποτελεί τη συλλογή εκατοντάδων τελικών συσκευών που συνδένται μεταξύ τους διαμέσου πυλών με υπηρεσίες και εφαρμογές του cloud. Για να επικοινωνήσουν οι συσκευές αυτές απαιτείται πλήρης διαλειτουργικότητα και παροχή συσκευών τέτοιων στην αγορά που να αναπτύσσει τις πλήρεις δυνατότητες του IoT (ARM White Paper, 2014). Οι επεξεργαστές της ARM αποτελούν μία οικογένεια 32-bit MCU οι οποίοι παρέχουν μία σταθερή βάση για το IoT. Βέβαια, η ανάπτυξη του λογισμικού για αυτές τις συσκευές στις μέρες μας είναι ιδιαίτερα ακριβή και αργή και συχνά χρησιμοποιεί εμπορικά λογισμικά και πρωτόκολλα επικοινωνίας.

Η ARM και οι συνεργάτες της οραματίζονται τη δημιουργία και την ανάπτυξη IoT συσκευών που θα είναι διαθέσιμες στο ευρύ κοινό και η πλατφόρμα του mbed είναι ένα αποτέλεσμα του οράματος αυτού (ARM White Paper, 2014).

Η πλατφόρμα αυτή έχει δημιουργηθεί με βάση τα ανοικτά πρότυπα και φέρνει τα πρωτόκολα του ίντερνετ, τα πρότυπα ασφάλειας και τη διαχείριση σε μία και μόνο ενσωματωμένη λύση που είναι βέλτιστη από άποψη διαχείρισης ενέργειας. Υποστηρίζεται από ένα οικοσύστημα εύκολα επεκτάσιμου υλικού και λογισμικού που παρέχει κατασκευαστικά block για συσκευές και υπηρεσίες του IoT.

Τα τρία βασικά μέρη του ARM mbed είναι το mbed OS, ο mbed Device Server, και τα mbed εργαλεία και υπηρεσιες. Το mbed OS είναι ένα νέο, δωρεάν λειτουργικό σύστημα για συσκευές που βασίζονται στον ARM Cortex-M0 που ενοποιεί τα θεμελιώδη κατασκευαστικά για IoT blocks (ARM White Paper, 2014). Είναι πρωτοποριακό σε θέματα ασφάλειας ενώ περιέχει χαρακτηριστικά διαχείρισης επικοινωνίας και συσκευών έτσι ώστε να οδηγήσει στην ανάπτυξη ενεργειακά αποδοτικών εφαρμογών για IoT. Το Mbed Os αποτελεί μία εξέλιξη του mbed SKD καθώς περιέχει μία επιπλέον καινοτόμα τεχνολογία. Προσφέρει μία κατευθυνόμενη από API προσέγγιση στον προγραμματισμό η οποία ελαχιστοποιεί το φόρτο που απαιτούνταν σε ότι αφορά την ανάπτυξη του MCU κώδικα (ARM White Paper, 2014).




**ΕΙΚΟΝΑ 15 : MBED OS**

Οι προγραμματιστές συνήθως γράφουν κώδικα χρησιμοποιώντας συναρτήσεις και κλήσεις API που έχουν ήδη δοκιμαστεί. Αυτό τους ελευθερώνει από την εστίαση στην προστιθέμενη αξία και δεν ανησυχούν σχετικά με την υλοποίηση του κώδικα στην MCU, τα περιφερειακά και το λειτουργικό σύστημα.

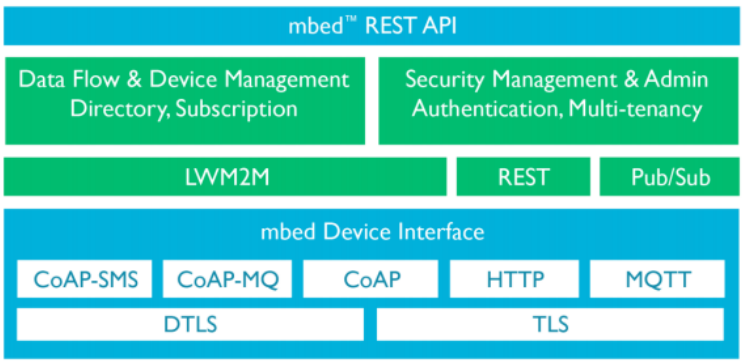
Η κατανάλωση της ισχύος είναι ένας από τους βασικούς περιορισμούς σε ότι αφορά τις IoT συσκευές. Πολλές από αυτές χρησιμοποιούν μπαταρίες ή πηγές ισχυρής κατανάλωσης ενέργειας οι οποίες δημιουργούν επιπλέον περιορισμούς στο υλικό, το λογισμικό και το πρωτόκολλο επικοινωνίας (ARM White Paper, 2014). Οι προγραμματιστές πρέπει να γνωρίζουν όλους τους σύνθετους τρόπους διαχείρισης και πολύ προσεκτικά να σχεδιάσουν αλγορίθμους μου ελαχιστοποιούν την κατανάλωση ενέργειας κατά την πρόσβαση στη μνήμη και την εναλλαγή κατάσταση. Επίσης, είναι ιδιαίτερα σημαντικό να διατηρείται η συσκευή σε κατάσταση ανενεργή όταν αυτή δεν ανιχνεύει το περιβάλλον ή δεν επικοινωνεί δεδομένα. Το mbed OS αντιμετωπίζει τις προκλήσεις αυτές υποστηρίζοντας ένα framework γεγονότων και αυτοματοποιημένη διαχείριση ισχύος παρέχοντας έτσι ένα αποτελεσματικό σε διαχείριση ισχύος τρόπο ώστε να συγγράφεται λογισμικό για τις περιορισμένες IoT συσκευές (ARM White Paper, 2014). Το κόστος των IoT συσκευών είναι πολύ σημαντικό λόγω της ευρείας κλίμακας ανάπτυξής τους. Άμεση λοιπόν συνέπεια της εξοικονόμησης του κόστους είναι ο περιορισμός σε υλικό. Το mbed Os έχει κατασκευαστεί έτσι ώστε να απαιτεί ελάχιστη μνήμη για να υποστηρίξει την ασφάλεια αλλά και τα πρωτόκολλα επικοινωνίας και διαχείρισης. Η ασφάλεια αποτελεί κυρίαρχο θέμα σε ότι αφορά στις εφαρμογές του IoT και την αφομοίωση αυτών από τους χρήστες (ARM White Paper, 2014).

Κάθε IoT σύστημα αποτελείται από τις συσκευές, το κανάλι επικοινωνίας και τη διαδικτυακή εφαρμογή. Για να είναι το σύστημα αυτό ασφαλές θα πρέπει οι συσκευές να εμπιστεύονται το κανάλι επικοινωνίας και αυτό με τη σειρά του να εμπιστεύεται τη διαδικτυακή εφαρμογή. Το mbed OS περιέχει όλα εκείνα τα στοιχεία που είναι απαραίτητα ώστε να υπάρχει ανάπτυξη ασφαλών και αποτελεσματικών IoT εφαρμογών (ARM White Paper, 2014). Περιλαμβάνει διάφορα στοιχεία ασφάλεια τους λογισμικού, ασφαλείς ενημερώσεις του firmware, κρυπτογράφηση και ασφαλή επικοινωνία.

Τα πρότυπα επιτρέπουν την αποδοχή ή τη συμβατότητα των τεχνολογιών καθώς όλα τα μέρη που συμμετέχουν το IoT δεν ανήκουν μονάχα σε έναν πάροχο. Τα ανοικτά πρότυπα, επιτρέπουν επίσης και τη διαλειτουργικότητα ανάμεσα στις διάφορες συσκευές στην αγορά. Το mbed OS υποστηρίζει όλα τα ανοικτά πρότυπα σύνδεση και διαχείρισης των συσκευών (ARM White Paper, 2014). Ακόμη, απαιτούνται διαφορετικές τεχνολογίες επικοινωνίας για διαφορετικές IoT εφαρμογές. Ανάλογα με το περιβάλλον στο οποίο υλοποιείται το σύστημα, η συσκευή μπορεί να έχει διαφορετικούς τύπους πρωτοκόλλων επικοινωνίας. Βασικά πρωτόκολλα επικοινωνίας όπως το Bluetooth® Smart, το Cellular, το Thread, WiFi®, και το 802.15.4/6LoWPAN σε συνδυασμό με το TLS/DTLS, το CoAP, το HTTP και το MQTT υποστηρίζονται από το mbed OS (ARM White Paper, 2014).

Ο Device Server είναι ένα λογισμικό που δίνει τη δυνατότητα σε τεχνολογίας στη μεριά του server να διαχειρίζονται και να ενώνουν συσκευές με έναν ασφαλή τρόπο. Λειτουργεί ως μία γέφυρα ανάμεσα σε πρωτόκολλα που σχεδιάζονται για να χρησιμοποιούνται σε IoT συσκευές και API. Όπως ένας web server δέχεται συνδέσεις από κινητά τηλέφωνα ή web browsers, ο Device Manager διαχειρίζεται τις συνδέσεις ανάμεσα σε IoT συσκευές. Ο mbed Device Manager αποτελεί ένα «κλειδί» για παρόχους υπηρεσιών cloud, λειτουργούς και επιχειρήσεις που έχουν πρόσβαση στην αγορά αυτή και προσφέρουν εμπορικές εφαρμογές.

Η επικοινωνίας στον mbed Device Server βασίζεται σε ανοικτά πρότυπα ενώ αυτός χρησιμοποιεί πρωτόκολλά όπως το CoAP, το HTTP, το MQTT, το TLS και το DTLS για τις συσκευές και τη διεπαφή RESTful για διαδικτυακές εφαρμογές. Το γεγονός αυτό εγγυάται μία IoT λύση που είναι σίγουρη, διαλειτουργική και εκμεταλλεύεται το τεράστιο οικοσύστημα ανάμεσα σε πωλητές, δημιουργούς υπηρεσιών και παρόχου υπηρεσιών (ARM White Paper, 2014).




**ΕΙΚΟΝΑ 16 : MBED DEVICE SERVER**

Το bandwidth και η αποτελεσματική διαχείριση ενέργειας είναι απαραίτητα για τη μείωση του κόστους και την κατανάλωση ισχύος. Ο mbed Device Server επιτρέπει την ελάχιστη χρήση εύρους ζώνης και διαθέσιμων πηγών μειώνοντας των αριθμό των αιτήσεων από τις διαδικτυακές εφαρμογές προς τις IoT συσκευές (ARM White Paper, 2014). Από τη στιγμή που όλες οι διαθέσιμες πηγές θα καταγραφούν στον κατάλογο των πηγών, η αναζήτηση, η ανάλυση και η ανακάλυψη μπορεί να είναι διαχειρίσιμη από τις διαδικτυακές εφαρμογές και τον mbed Device Server. Επιπλέον, τα δεδομένα που απαιτούνται από τη διαδικτυακή εφαρμογή μπορούν να αποκτηθούν από την cache των πηγών.

Επιπλέον, για τις συσκευές που παρουσιάζουν περιορισμούς, το CoAP μειώνει το εύρος ζώνης κατά 10x εάν αυτό συγκριθεί με το HTTP. Έτσι, απαιτείται απλοποιημένη ανάπτυξη για γρήγορη τοποθέτηση στην αγορά. mbed Device Server παρέχει ένα απλό REST interface το οποίο διευκολύνει την γρήγορη ανάπτυξη των web εφαρμογών. Επιπλέον, περιλαμβάνει ένα Java SDK για τις εφαρμογές που αφορούν στους αυτοματισμούς σπιτιού και στον έξυπνο φωτισμό. Δεν χρειάζεται κάποια ειδική γνώση σε ότι αφορά την ανάπτυξη των ιστοσελίδων για να χρησιμοποιηθούν τα παραπάνω. Χρειάζονται χαρακτηριστικά κλιμάκωσης ώστε χιλιάδες συσκευές να μπορούν να συνδεθούν με μία IoT εφαρμογή. Ο mbed Device Server υποστηρίζει εξισορρόπηση φόρτου και clustering έτσι ώστε να ενισχύσει την κλιμάκωση (ARM White Paper, 2014). Η ομαδοποίηση των τελικών συσκευών επίσης υποστηρίζεται από batch λειτουργίες. Τα χαρακτηριστικά αυτά καθιστούν ευκολότερη την ανάπτυξη διαδικτυακών εφαρμογών και βελτιώνουν την αποτελεσματικότητα του συστήματος.

Η ασφάλεια είναι ένας ιδιαίτερα σοβαρός παράγοντας στην αγορά αυτή και την αφομοίωση των IoT εφαρμογών. Ο mbed Device Server χρησιμοποιεί κρυπτογραφία δημόσιου κλειδιού και ένα καινοτόμο επίπεδο μεταφορά για την ασφάλεια των δεδομένων. Η ασφάλεια ανάμεσα στις IoT συσκευές και τον mbed Device Server παρέχεται με χρήση του DTLS πρωτοκόλλου για αμοιβαία αυθεντικοποίηση, προστασία της ακεραιότητας και εμπιστευτικότητα για όλη την CoAP κίνηση ανάμεσα στη συσκευή και την πλατφόρμα.

Η ασφάλεια ανάμεσα στις διαδικτυακές εφαρμογές και την πλατφόρμα γίνεται με χρήση του TLS για όλη την HTTP κίνηση (HTTPS) (ARM White Paper, 2014).

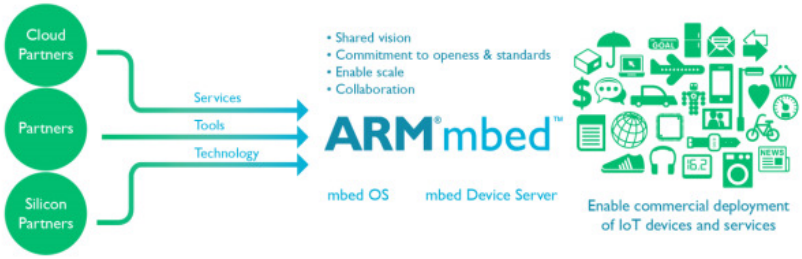
Ο mbed παρέχει επίσης εργαλεία και υπηρεσίες ώστε να επιταχύνει επιπλέον την ανάπτυξη. Ο συνδυασμός του υλικού με το λογισμικό κάνει εύκολη την ανακάλυψη πολλών και νέων σχεδιασμών. Παρέχεται ένα ιδιαίτερα ισχυρό IDE που είναι ελεύθερο στο να χρησιμοποιηθεί και είναι πλήρως ενσωματωμένο στο mbes OS. Ακόμη, δίνεται η δυνατότητα χρήσης ενός SDK με εργαλεία που περιλαμβάνουν υποστήριξη της Keil, της IAR και του GCC (ARM White Paper, 2014). Το online IDE δίνει τη δυνατότητα της άμεσης εκκίνησης χωρίς να χρειάζεται κάποια επιπλέον εγκατάσταση. Ακόμη, περιλαμβάνει κάποιον code editor με πολλαπλά αρχεία, φακέλους, προγράμματα αλλά και ένα drag and drop interface. Επιπλέον, περιλαμβάνει χαρακτηριστικά όπως την αναζήτηση ανάμεσα σε πολλαπλά αρχεία και τύπους αρχείων. Η ενσωματωμένη έκδοση επιτρέχει την συγχώνευση κώδικα αλλά δίνει και μία αναπαράσταση όλων των προηγούμενων εκδόσεων αυτού (ARM White Paper, 2014). Η εισαγωγή βιβλιοθηκών ή παραδειγμάτων διαμέσου του οδηγού εισαγωγής δίνει τη δυνατότητα εισαγωγής βιβλιοθηκών ή προγραμμάτων που έχουν κατασκευαστεί από άλλους χρήστες. Αυτό είναι ιδιαίτερα χρήσιμο σε ότι αφορά στην εισαγωγή κώδικα και στη δημιουργία block κώδικα.

Το online IDE χρησιμοποιεί τον compiler ARM RVDS 4.1 ο οποίος είναι βελτιστοποιημένος και δίνει ένα πολύ καλό μέγεθος στον κώδικα ενώ μεγαλώνει την απόδοση (ARM White Paper, 2014). Δεν υπάρχει κανένας περιορισμός σχετικά με το μέγεθος του κώδικα, ενώ ο κώδικας που παράγεται μπορεί να χρησιμοποιηθεί για εμπορικούς και μη σκοπούς.

Το ARM mbed Partner οικοσύστημα ενώνει την ενσωματωμένη τεχνολογία με το cloud αλλά και με τους κατασκευαστές των εξοπλισμών, τους ανθρώπους που

υλοποιούν τα συστήματα αλλά και όλους αυτούς που θέλουν να κάνουν χρήση της τεχνολογίας, των εργαλείων και των υπηρεσιών που χρειάζονται ώστε να επιταχυνθεί η εξέλιξη της καινοτομίας στη δημιουργία και την ανάπτυξη IoT συστημάτων (ARM White Paper, 2014).

Τα μέλη του οικοσυστήματος αυτού διαμοιράζονται ένα όραμα για το μέλλον όπου η ανάπτυξη και η εφαρμογή των εμπορικών IoT συσκευών θα έχει τη δυνατότητα κλιμάκωσης, τη δυνατότητα δέσμευσης σε ανοικτότητα και πρότυπα και μία επιθυμία συνεργασίας σε σχέδια και project που θα κάνουν το όραμα αυτό πραγματικότητα.




**ΕΙΚΟΝΑ 17 : ΤΟ ARM MBED ΟΙΚΟΣΥΣΤΗΜΑ**

## 3.5 ΕΠΙΛΟΓΗ ΑΝΑΠΤΥΞΙΑΚΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

Για τον προγραμματισμό της πλακέτας, χρησιμοποιήσαμε διάφορα εργαλεία τα οποία και αναφέραμε πιο πάνω. Οι λόγοι που επιχειρήσαμε να πειραματιστούμε με όλα τα εργαλεία ήταν δύο. Αρχικά, για να βρούμε το πιο εύχρηστο αναπτυξιακό περιβάλλον που θα μας επέτρεπε να εκμεταλλευτούμε στο μέγιστο τις δυνατότητες του επεξεργαστή. Επιπλέον, θέλαμε να βρούμε ένα αρκετά απλό, στην εκμάθηση, τρόπο για να μπορούν μελλοντικοί φοιτητές να ξεκινήσουν να ασχολούνται με επεξεργαστές ARM. Μετά λοιπόν από πειραματισμούς, καταλήξαμε ότι, τα δύο ιδανικότερα περιβάλλοντα είναι το μVision της KEIL και ο online compiler ARM MBED, σύμφωνα πάντα με τη δική μας άποψη. Το περιβάλλον της KEIL μVision 4 είναι ένα αναπτυξιακό λογισμικό που βασίζεται σε παράθυρα και παρέχει μια πληθώρα από χρήσιμες βιβλιοθήκες και compilers, που επιτρέπουν την δημιουργία συμπυκνωμένου κώδικα. Επίσης, μας δώθηκε η δυνατότητα της εξομοίωσης του συστήματος από τον υπολογιστή, έτσι ώστε να γίνεται εύκολα και γρήγορα η αποσφαλμάτωση, ακόμη και χωρίς το αντίστοιχο hardware. Μπορούμε λοιπόν να επιλέξουμε μεταξύ Simulator Mode και Target Mode, ανάλογα με το αν θέλουμε να κάνουμε μια εξομοίωση η να

τρέξουμε τον κώδικα στο σύστημά μας. Στο Simulator Mode το μVision δίνει την δυνατότητα να ελέγχουμε κάποιες παραμέτρους που στο hardware θα ήταν αδύνατο όπως το να θέσουμε breakpoints σε σημεία που δεν γίνεται με το hardware. Η online πλατφόρμα ARM MBED ήταν εξίσου πάρα πολύ χρήσιμη. Με ένα απλό log in μπορείς να βρεις αμέτρητους έτοιμους κώδικες από άλλους χρήστες και να πειραματιστείς στο δικό σου περιβάλλον. Δεν έχει πολλά παράθυρα οπως το μVision της KEIL. Είναι ένα εργαλείο που κάνεις αυτό που λέμε λίγο πιο πρόχειρη δουλειά μέχρι να καταλήξεις στο τελικό σου πρόγραμμα. Μας βοήθησε πάρα πολύ. Το Codewarrior της Freescale μοιάζει αρκετά με το μVision αλλά κατά τη γνώμη μας είναι αρκετά πιο πολύπλοκο. Για κάποιο λόγο δυσκολευτήκαμε αρκετά στο να φορτώσουμε τους κώδικες μας στην πλακέτα μας. Επίσης το debugging mode του είναι δύσκολο ως προς την κατανόηση αλλά και χρήση του. Εμείς προτείνουμε ανεπιφύλακτα το προγραμματιστικό περιβαλλον της KEIL, μVision 4.

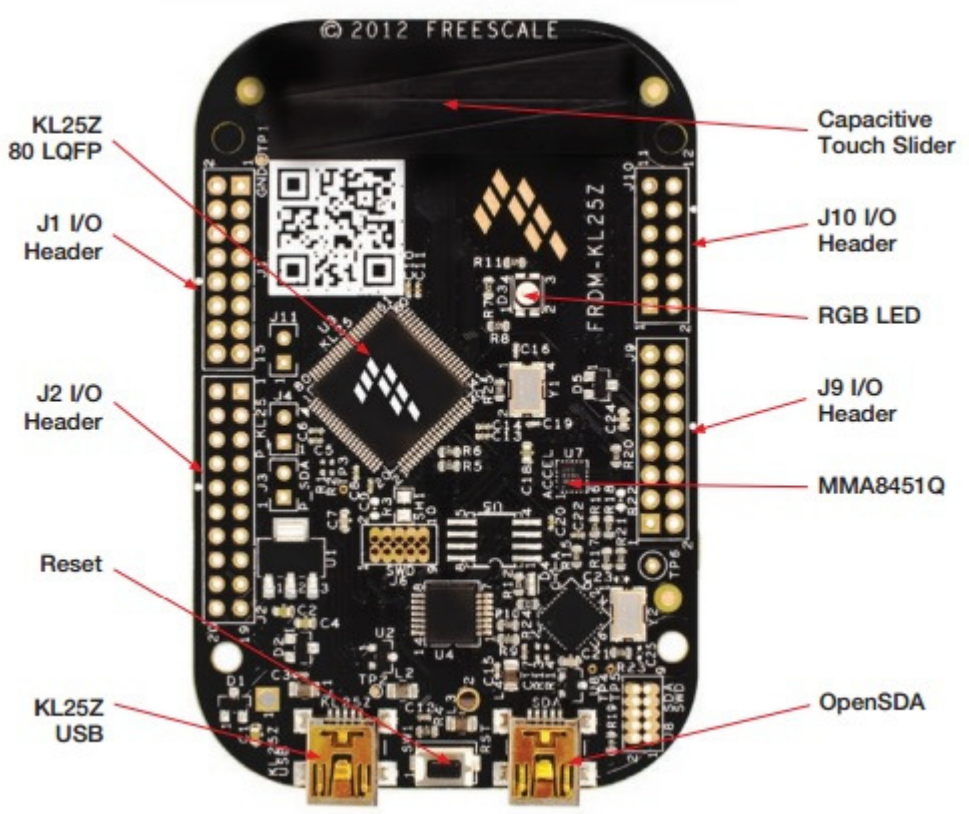
# ΚΕΦΑΛΑΙΟ 4$^{Ο}$ – ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΣΙΑΣ

## 4.1 ΠΕΡΙΓΡΑΦΗ ΠΛΑΤΦΟΡΜΑΣ ΚΑΙ ΑΜΠΕΡΟΜΕΤΡΟΥ

Η πλατφόρμα FRDM KL25Z της FREESCALE είναι ένα μικρό, χαμηλής κατανάλωσης και χαμηλού κόστους αναπτυξιακό σύστημα για γρήγορο προγραμματισμό πρωτότυπων εφαρμογών.

Κάθε μία από αυτές τις πλατφόρμες είναι εύκολη στη χρήση και στον προγραμματισμό τους καθώς διαθέτουν θύρα USB, κάτι που καθιστά εύκολη την σειριακή επικοινωνία με τον υπολογιστή.

Είναι πολύ εύκολο να ξεκινήσει κανείς. Πολύ απλά επιλέγεις ένα από τα πολλά περιβάλλοντα ανάπτυξης και ξεκινάς τον προγραμματισμό της πλατφόρμας.



## Χαρακτηριστικά

Η πλακέτα FRDM-KL25Z περιλαμβάνει έναν 32-bit ARM Cortex-M0+ επεξεργαστή,ο οποίος τρέχει στα 48 MHz.Επίσης, περιέχει μια flash μνήμη 128 KB,μια RAM μνήμη 16KB καθώς και αρκετά interfaces όπως USB Host, USB Device, SPI, I2C, ADC, DAC, PWM, Touch Sensor,GPIO.

Τα τρία βασικά components της πλατφόρμας μας είναι το

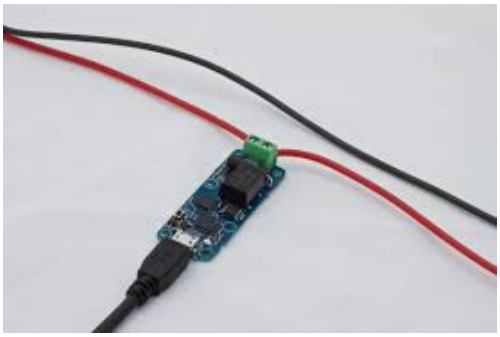
- MMA8451Q - 3-axis accelerometer(επιταχυνσιόμετρο)
- PWM Controlled RGB LED
- Capacitive touch sensor(αισθητήρας αφής)

Αξίζει να σημειώσουμε ότι για την επικοινωνία των δεδομένων πρέπει να χρησιμοποιηθεί ο connector KL25Z USB και όχι το OpenSDA!

Επίσης, μια άλλη συσκευή που χρησιμοποιήσαμε για να κάνουμε τις μετρήσεις ήταν το YoctoAmperometer. Η συσκευή αυτή περιλαμβάνει ένα ψηφιακό USB το οποίο μετράει αυτόματα ρεύμα. Λειτουργεί επίσης είτε με συνεχές ρεύμα είτε με εναλλασσόμμενο. Το αμπερόμετρο αυτό ειναι απομωνομένο με την έννοια ότι το κομμάτι του αισθητήρα είναι ηλεκτρικά αποσυνδεδεμένο από το μέρος του USB και έτσι δεν υπάρχει κανένας φόβος να “καεί” το PC μας. Τα χαρακτηριστικά του είναι τα εξής:

- Πλάτος – 20 mm
- Μήκος – 56 mm
- Βάρος – 8 gr
- Micro USB connector

- Max current (Cont.) -- 10A ///// Max current (peak) -- 17A



## 4.2 ΠΕΡΙΓΡΑΦΗ ΤΩΝ POWERSTAGES ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ

Ο ελεγκτής του συστήματος (SMC) είναι υπεύθυνος για να κάνει όλες εκείνες τις μεταβιβάσεις από τη μία κατάσταση ( powerstage ) στην άλλη. Αυτό το πετυχαίνει κυρίως ελέγχοντας τα ρολόγια(clocks) και τις μνήμες(memories). Σ'αυτό το κεφαλαιο θα σας περιγράψουμε όλες τις καταστάσεις που μπορεί να εισέλθει ένας επεξεργαστής σαν το δικό μας.

Ο ARM επεξεργαστής(CPU) διαθέτει 3 βασικές κλίμακες λειτουργίας:

- RUN
- Sleep
- Deep Sleep

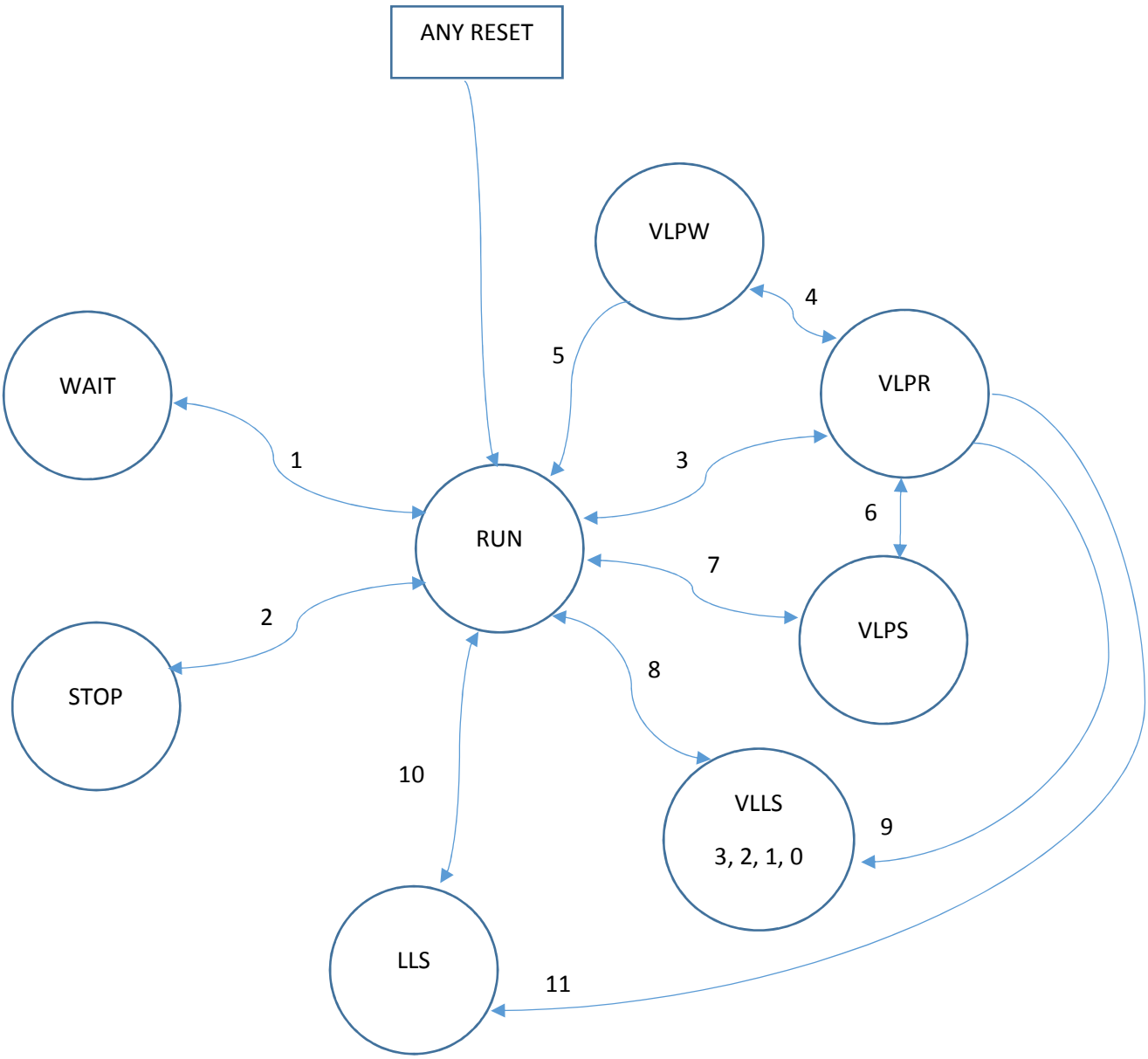
Αυτά ισχύουν συγκεκριμένα για τους ARM επεξεργαστές. Γενικότερα,για τους μικροεπεξεργαστές (MicroControllersUnits) έχουμε :

- RUN
- WAIT
- STOP

Στον παρακάτω πίνακα θα κάνουμε μια μικρή περιγραφή όλων των καταστάσεων (power modes) της συσκευής μας.

| Κατάσταση | Περιγραφη |
|---|---|
| RUN | Ο μικροεπεξεργαστής μπορεί να τρέχει στη μέγιστη ταχύτητα και η τροφοδοσία ειναι πλήρως ρυθμιζόμενη. Αυτή η κατάσταση είναι γνωστή και ως Normal Run Mode. |
| WAIT | Το ρολόι του πυρήνα απενεργοποιείται. Το ρολόι του συστήματος συνεχίζει να λειτουργεί. Τα ρολόγια των διάυλων, αν είναι ενεργά, συνεχίζουν να λειτουργούν. |
| STOP | Το ρολόι του πυρήνα απενεργοποιείται. Τα ρολόγια του συστήματος και των διάυλων απενεργοποιούντε. |
| VLPR (VeryLowPowerRun) | Οι μέγιστες συχνότητες των ρολογιών πυρήνα, διάυλων και Flash είναι περιορισμένες σε αυτή την κατάσταση. |
| VLPW (VeryLowPowerWait) | Το ρολόι του πυρήνα απενεργοποιείται. Τα ρολόγια πυρήνα, διάυλων και flash συνεχίζουν να λειτουργούν αλλά οι μέγιστες συχνότητές τους είναι περιορισμένες. |
| VLPS (VeryLowPowerStop) | Το ρολόι του πυρήνα απενεργοποιείται. . Τα ρολόγια του συστήματος και των διάυλων απενεργοποιούντε. Ο μικροεπεξεργαστής μπαίνει σε κατάσταση χαμηλής κατανάλωσης μειώνοντας την ταση. |
| LLS (LowLeakageStop) | Το ρολόι του πυρήνα απενεργοποιείται. Τα ρολόγια του συστήματος και των διάυλων απενεργοποιούντε. Ο μικροεπεξεργαστής μπαίνει σε κατάσταση χαμηλής κατανάλωσης μειώνοντας την ταση “to internal logic”. |
| VLLS3 (VeryLowLeakageStop) | Το ρολόι του πυρήνα απενεργοποιείται. Τα ρολόγια του συστήματος και των διάυλων απενεργοποιούντε. Ο μικροεπεξεργαστής μπαίνει σε κατάσταση χαμηλής κατανάλωσης απενεργοποιόντας “the internal logic”. |

## Σχεδιάγραμμα απεικόνισης μεταβιβάσεων



## Πίνακας Μεταβιβάσεων

| Transition # | From | To | Trigger conditions |
|---|---|---|---|
| **1** | RUN | WAIT | Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, controlled in System Control Register in ARM core. |
| | WAIT | RUN | Interrupt or Reser |
| **2** | RUN | STOP | PMCTRL[RUNM]=00, PMCTRL[STOPM]=000[2]<br>Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1] |
| | STOP | RUN | Interrupt or Reset |
| **3** | RUN | VLPR | The core, system, bus and flash clock frequencies are restricted in this mode. See the Power Management chapter for the maximum allowable frequencies.<br>Set PMPROT[AVLP]=1, PMCTRL[RUNM]=10. |
| | VLPR | RUN | Set PMCTRL[RUNM]=00 or<br>Reset. |
| **4** | VLPR | VLPW | Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, which is controlled in System Control Register in ARM core. See note.[1] |
| | VLPW | VLPR | Interrupt |
| **5** | VLPW | RUN | Reset |
| **6** | VLPR | VLPS | PMCTRL[STOPM]=000[3] or 010,<br>Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1] |
| | VLPS | VLPR | Interrupt<br>**NOTE:** If VLPS was entered directly from RUN, hardware will not allow this transition and will force exit back to RUN |
| **7** | RUN | VLPS | PMPROT[AVLP]=1, PMCTRL[STOPM]=010,<br>Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1] |
| | VLPS | RUN | Interrupt and VLPS mode was entered directly from RUN or<br>Reset |

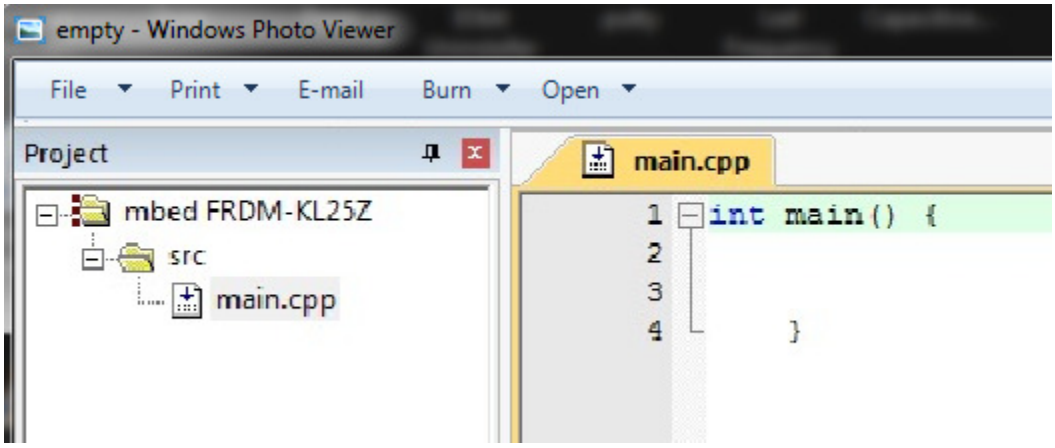
| | | | |
|---|---|---|---|
| **8** | RUN | VLLSx | PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[VLLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| | VLLSx | RUN | Wakeup from enabled LLWU input source or RESET pin |
| **9** | VLPR | VLLSx | PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[VLLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| **10** | RUN | LLS | PMPROT[ALLS]=1, PMCTRL[STOPM]=011, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |
| | LLS | RUN | Wakeup from enabled LLWU input source or RESET pin. |
| **11** | VLPR | LLS | PMPROT[ALLS]=1, PMCTRL[STOPM]=011, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. |

Στον πίνακα μεταβιβάσεων φαίνεται αναλυτικά πως μπορούμε να πάμε απο τη μια κατάσταση στην άλλη. Υπάρχουν συγκεκριμένες εντολές μεταβίβασης και αρκεί μόνο ένα απλό reset έτσι ώστε να επαναφέρουμε την πλακέτα μας στην αρχική της κατάσταση.

## 4.3 ΤΡΟΠΟΣ ΥΛΟΠΟΙΗΣΗΣ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΚΑΤΑΝΑΛΩΣΗΣ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ

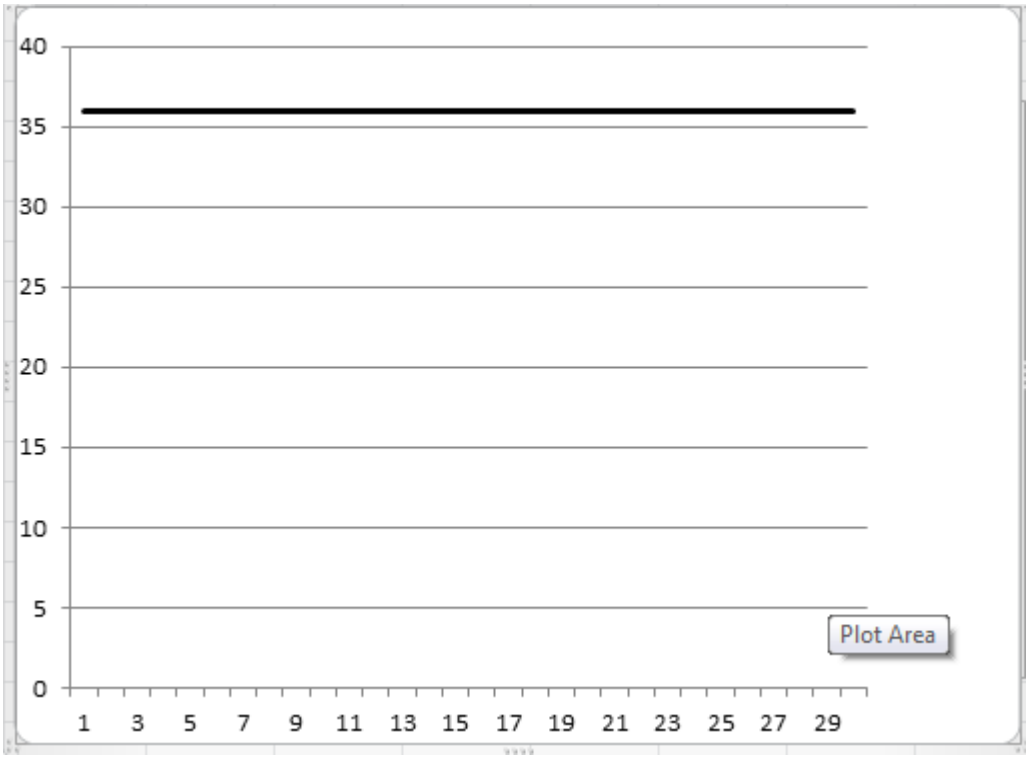
### 4.3.1 ΚΑΤΑΝΑΛΩΣΗ ΕΝΕΡΓΕΙΑΣ ARM ΣΕ ΑΔΡΑΝΗ ΚΑΤΑΣΤΑΣΗ

Σ΄αυτό το κεφάλαιο θα δούμε αναλυτικά τα προγράμματα που φτιάξαμε, τις μετρήσεις που πήραμε με βάση την κατανάλωση ισχύος του κάθε προγράμματος καθώς και τις κυματομορφές τους. Στην αρχή φτιάξαμε ένα «κενό» πρόγραμμα, χωρίς καμία λειτουργία, έτσι ώστε να δούμε την κατανάλωση του επεξεργαστή.




**ΕΙΚΟΝΑ 18 : EMPTY PROGRAM**

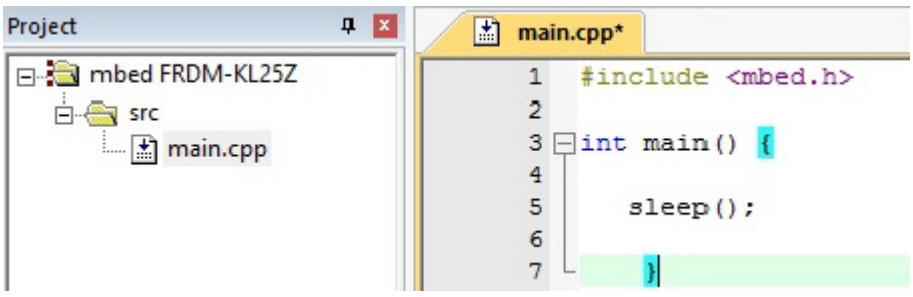
Η κατανάλωση που μας έβγαλε το αμπερόμετρο σε χρόνο 30 δευτερολέπτων ήταν σταθερή στα 36 mA



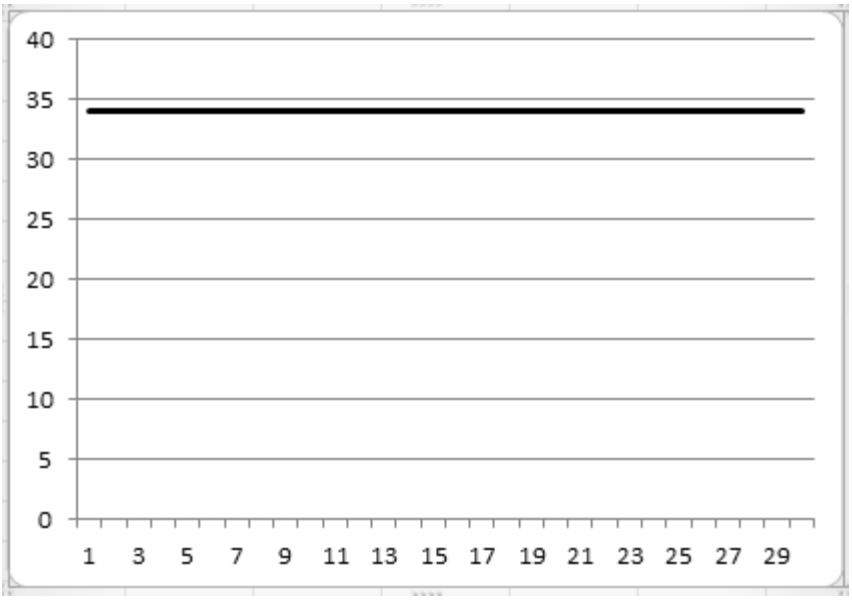

**ΕΙΚΟΝΑ 19 : POWER CONSUMPTION OF EMPTY PROGRAM**

¨Έπειτα, μέσα σε αυτό το άδειο πρόγραμμα, προκειμένου να ρίξουμε την κατανάλωση κάτω απο τα 36 mA, βάλαμε μέσα στη main την εντόλή sleep().

Πράγματι, απο εκεί που είχαμε ένα πρόγραμμα σε active κατάσταση καταφέραμε και μπήκαμε σε sleep mode. Μετρήσαμε πάλι σε χρόνο 30 δευτερολέπτων και η κατανάλωση μας έπεσε στα 34 mA. Υπενθυμίζουμε ότι δε βλέπουμε μεγάλη διαφορά πτώσης του ρεύματος διότι η πλακέτα μας είναι μικρής κατανάλωσης(ultra low power microcontroller)

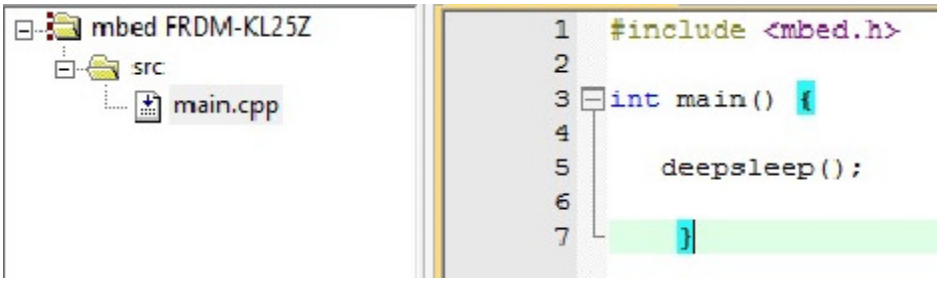



**ΕΙΚΟΝΑ 20 : SLEEP MODE**



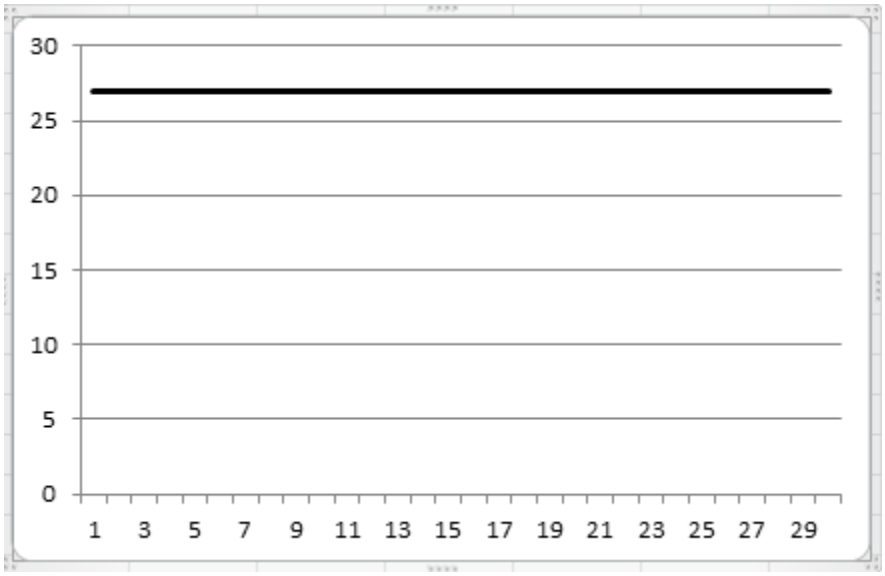

**ΕΙΚΟΝΑ 21 : POWER CONSUMPTION OF SLEEP MODE**

Γνωρίζοντας ότι η κατανάλωση είναι εφικτό να « πέσει » κ΄άλλο, έπρεπε να βρούμε μία εντολή έτσι ώστε να μεταβούμε σε deepsleep mode. Αυτό έγινε πολύ εύκολα με την εντολή deepsleep() όπως θα δείτε και στα screenshots.



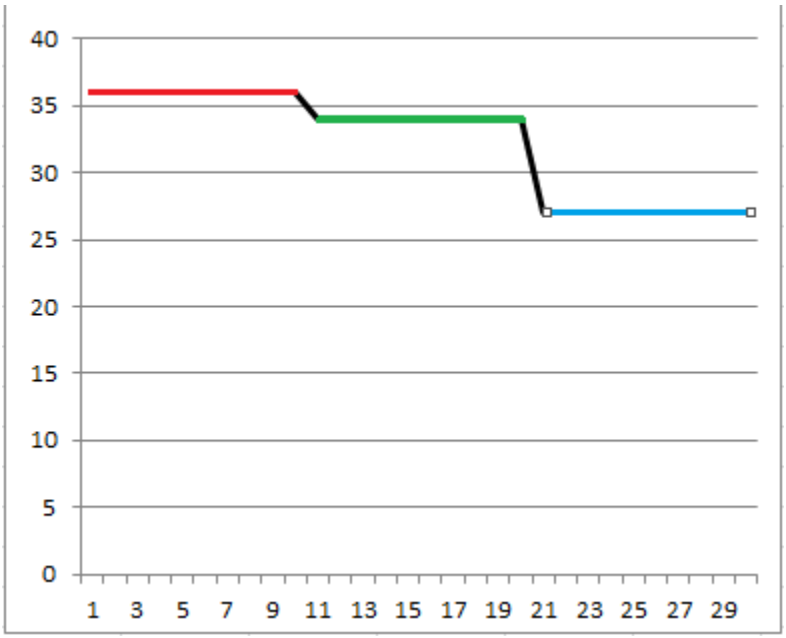

**ΕΙΚΟΝΑ 22 : DEEP SLEEP MODE**

Μετρήσαμε πάλι με το αμπερόμετρο για 30 δευτερόλεπτα και η τάση από τα 34 mA που ήταν σε sleep κατάσταση, έπεσε στα 27 mA




**ΕΙΚΟΝΑ 23: POWER CONSUMPTION OF DEEP SLEEP MODE**

Ας δούμε τώρα σε έναν πίνακα και τις τρεις αυτές καταστάσεις του επεξεργαστή μαζί.



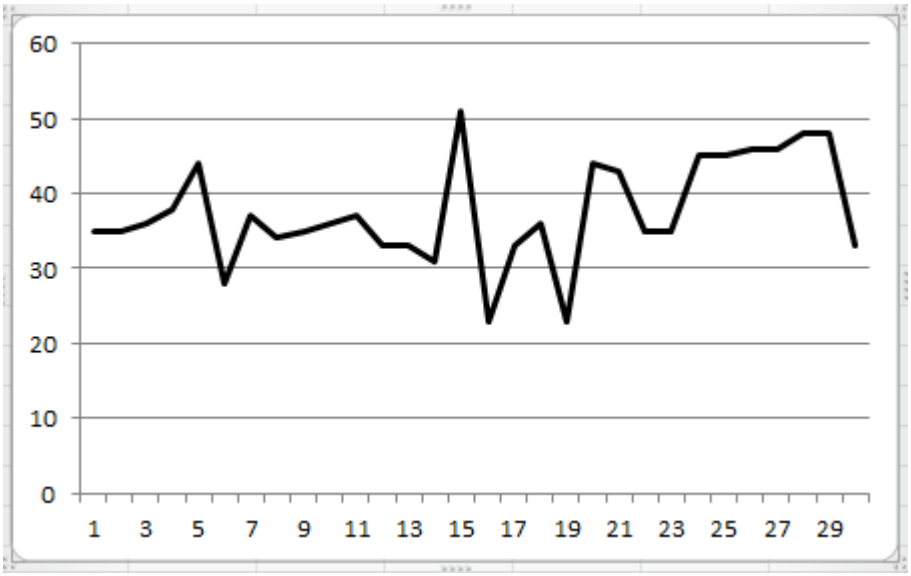

**ΕΙΚΟΝΑ 24 : POWER CONSUMPTION OF ACTIVE-SLEEP-DEEPSLEEP MODE**

Οπως φαίνεται καθαρά στον παραπάνω πίνακα στα πρώτα δέκα δευτερόλεπτα έχουμε το κενό πρόγραμμα μας του οποίου ο επεξεργαστής καταναλώνει 36 mA ρεύμα(κόκκινη γραμμή-active mode). Στη συνέχεια, σε κλάσματα δευτερολέπτου

έχουμε την πτώση της τάσης στα 34 mA για τα επόμενα δέκα δεύτερα(πράσινη γραμμή-sleep mode). Τέλος, πάλι μέσα σε ελάχιστα msec η κατανάλωση πέφτει στα 27 mA(μπλε γραμμή-deepsleep mode) εως ότου σταματάμε την μέτρηση στα 30 δεύτερα. Αν θέλουμε να επαναφέρουμε το πρόγραμμα σε active mode, αρκεί ένα απλό reset.

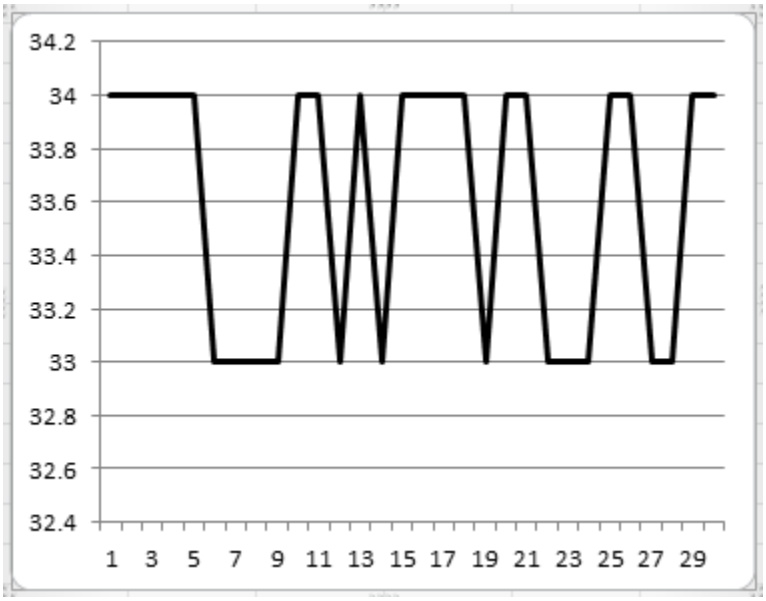
### 4.3.2 ΚΑΤΑΝΑΛΩΣΗ ΑΙΣΘΗΤΗΡΑ ΕΠΙΤΑΧΥΝΣΗΣ(accelerometer)

Το επόμενο πρόγραμμα που φτιάξαμε έχει να κάνει με το επιταχυνσιόμετρο (accelerometer). Στην ουσία δημιουργήσαμε τρεις άξονες (x, y, z) πάνω στους οποίους κινούμε την πλακέτα μας και επιτυγχάνουμε αλλαγή χρώματος του Led. Στον άξονα «X» έχουμε δηλώσει το μπλε χρώμα, στον άξονα «Y» το κόκκινο και στον «Z» το πράσινο. Παρατηρήσαμε ότι κινώντας την πλακέτα μας ΜΟΝΟ στον οριζόντιο άξονα(μπλε) είχαμε μια κατανάλωση 35mA περίπου. Την ίδια κατανάλωση είχαμε και με κίνηση ΜΟΝΟ πάνω στον κάθετο άξονα(κόκκινο) αλλά και στον κατακόρυφο(πράσινο). Με τυχαία κίνηση πάνω και στους τρεις άξονες είχαμε το εξής αποτέλεσμα:




**ΕΙΚΟΝΑ 25 : POWER CONSUMPTION OF ACCELEROMETER**

Το αποτέλεσμα είναι απολύτως φυσιολογικό διότι με την τυχαία κίνηση της πλακέτας μας πάνω από τους άξονες, το LED παίρνει όλα τα χρώματα(r-g-b) αλλά και συνδυασμό αυτών με αποτέλεσμα τις αυξομειώσεις της κατανάλωσης. Θα είχε ιδιαίτερο ενδιαφέρον να δούμε και την κατανάλωση ρεύματος του επιταχυνσιομέτρου με απενεργοποιημένα όμως τα leds.




**ΕΙΚΟΝΑ 26 : POWER CONSUMPTION OF ACCELEROMETER WITHOUT LEDS**

Παρατηρούμε ότι η κατανάλωση κυμαίνεται από 33mA έως 34mA .Τα δυο γραφήματα κατανάλωσης έχουν σημαντικές διαφορές και καταλήγουμε στο συμπέρασμα ότι τα leds δίνουν επιπλέον κατανάλωση στο ρεύμα μας.

```
#include "mbed.h"
#include "MMA8451Q.h"

PinName const SDA = PTE25;
PinName const SCL = PTE24;

#define MMA8451_I2C_ADDRESS (0x1d<<1)

int main(void) {
   MMA8451Q acc(SDA, SCL, MMA8451_I2C_ADDRESS);
   PwmOut rled(LED1);
   PwmOut gled(LED2);
   PwmOut bled(LED3);

   printf("MMA8451 ID: %d\n", acc.getWhoAmI());

   while (true)
   {
      float x, y, z;
      x = rled = 1.0 - abs(acc.getAccX());
      y = gled = 1.0 - abs(acc.getAccY());
      z = bled = 1.0 - abs(acc.getAccZ());
      wait(0.1);
      printf("X: %1.2f, Y: %1.2f, Z: %1.2f\n", x, y, z);
   }
}
```

**ΕΙΚΟΝΑ 27 : ACCELEROMETER PROGRAM**

## 4.3.3 ΚΑΤΑΝΑΛΩΣΗ ΑΙΣΘΗΤΗΡΑ ΟΛΙΣΘΗΣΗΣ (slide bar)

Το αμέσως επόμενο πρόγραμμα που φτιάξαμε ονομάζεται touchbutton. Στην ουσία έχουμε χωρίσει το slider της πλακέτας σε τέσσερα κουμπιά που το κάθε έχει τη δική του λειτουργία. Πατώντας το πρώτο κουμπί ανάβει το μπλε led και εκτελείται μια πρόσθεση. Πατώντας το δεύτερο κουμπί ανάβει το πράσινο led και εκτελείται πολλαπλασιασμός(δύναμη του 2). Με το πάτημα του τρίτου κουμπιού ανάβει το κόκκινο led και εκτελείται μια διαίρεση. Στο τελευταίο κουμπί(άσπρο led) έχουμε την σειριακή επικοινωνία του υπολογιστή με τον χρήστη(serial communication). Δηλαδή, πατώντας το $4^{ο}$ κουμπί εμφανίζεται ένα μήνυμα σε ένα τερματικό που μας λέει:

”press g for green, b for blue, r for red and w for white”

Αν πατήσουμε οποιοδήποτε άλλο γράμμα μας βγάζει ERROR!

Από τις μετρήσεις που πήραμε παρατηρήσαμε ότι μεταξύ των πράξεων δεν υπάρχουν μεγάλες διαφορές στην κατανάλωση ισχύος. Εκεί που είδαμε μια αύξηση ήταν στο

πάτημα του τέταρτου κουμπιού λόγω του άσπρου χρώματος.Στα παρακάτω screenshots φαίνονται τα προγράμματα μας αναλυτικά.

```
      if(key==1)
{
     myled1 = 1;
     myled2 = 1;
     myled3 = 0;

    for (i=0; i<50; i++)
{
  a=a+5;
     pc.printf("%d\n",a );
     wait(0.3);
}
```

**ΕΙΚΟΝΑ 28 : BLUE LED-ADDITION**

```
else if(key==2)
{

     myled1 = 1;
     myled2 = 0;
     myled3 = 1;
  for (i=0; i<20; i++)
  {
    b=b*2;
    pc.printf("%d\n",b );
    wait(0.3);

  }
```

**ΕΙΚΟΝΑ 29 : GREEN LED-MULTIPLICATION**

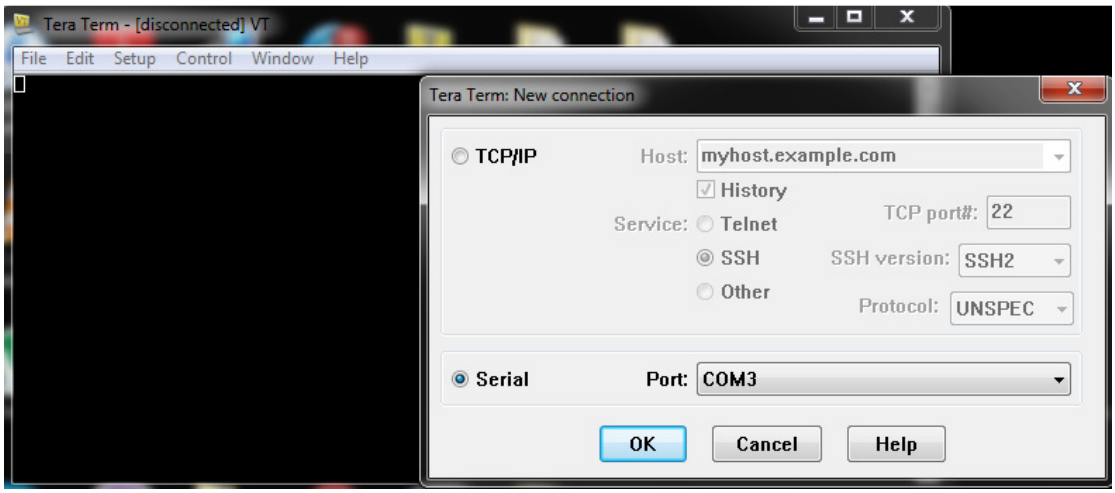
```
else if(key==3)
{
     myled1 = 0;
     myled2 = 1;
     myled3 = 1;
  for (i=0; i<20; i++)
  {
    c=c/2;
    pc.printf("%d\n",c );
    wait(0.3);

  }
```

**ΕΙΚΟΝΑ 30 : RED LED-DIVISION**

```
    else
    {
        myled1 = 1;
        myled2 = 1;
        myled3 = 1;
      pc.printf ("Press 'G' for Green Led, 'R' for Red Led, 'B' for Blue Led and 'W' for White Led\n");
      while(1) {
    char c = pc.getc();
    if(c == 'g') {
       myled1 = 1;
        myled2 = 0;
        myled3 = 1;
      pc.printf("GREEN\n");
    }
  else  if(c == 'r') {
        myled1 = 1;
        myled2 = 1;
        myled3 = 0;
    pc.printf("RED\n");
```

**ΕΙΚΟΝΑ 31 : WHITE LED-SERIAL COMMUNICATION BETWEEN USER AND PC**

Στις παρακάτω εικόνες φαίνονται τα αποτελέσματα από την σειριακή επικοινωνία μεταξύ του χρήστη και του υπολογιστή. Αυτό το πετύχαμε με τη χρήση ενός εικονικού τερματικού που ονομάζεται Tera Term. Στην ουσία είναι ένα λογισμικό ανοικτού-ελεύθερου κώδικα και χρησιμοποιείται για εξομοιώσεις επικοινωνίας.




**Εικόνα 32 : TERA TERMINAL**

Στις εικόνες από κάτω φαίνονται τα αποτελέσματα από τις πράξεις που κάναμε με το πρόγραμμα touch button. Στο μπλε led προστίθεται ο αριθμός 5 μέχρι να φτάσει στο εκατό. Στο πράσινο led έχουμε πολλαπλασιασμό(στην ουσία δύναμη του 2) και στο κόκκινο led κάνουμε μια διαίρεση όπως φαίνεται και παρακάτω. Με το πάτημα του λευκού led  φαίνεται ξεκάθαρα η άμεση επικοινωνία υπολογιστή και χρήστη. Πατάμε το “g”, ανάβει το πράσινο led. Πατάμε το “ b”, ανάβει το μπλε led κλπ. ΑΝ πατήσουμε οτιδήποτε άλλο, μας εμφανίζει μήνυμα λάθους.

```
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
```

**Blue led - Add**

```
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
```

**Green led - multiplication**

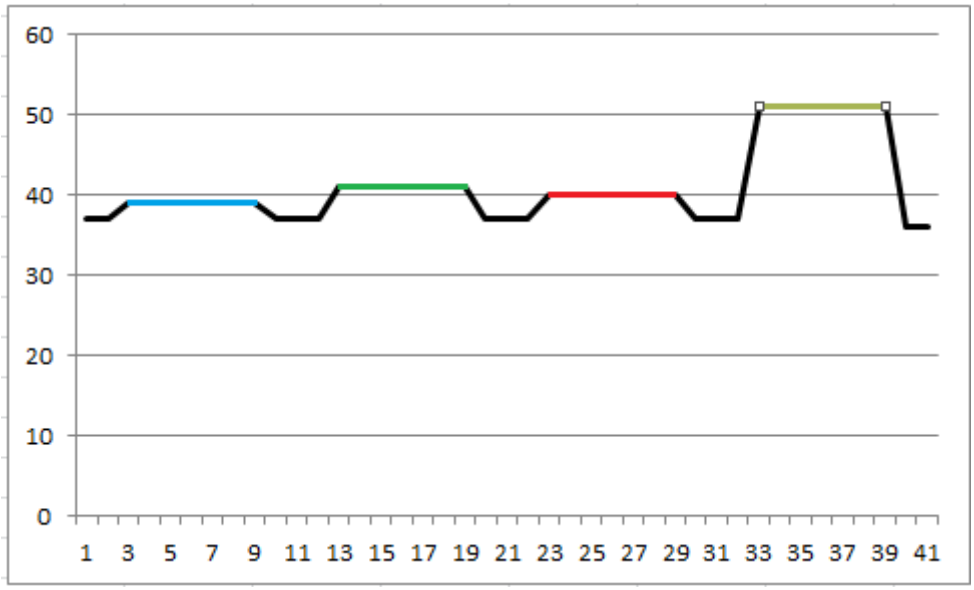
```
500000
250000
125000
62500
31250
15625
7812
3906
1953
976
488
```

**Red led - division**

```
Press 'G' for Green Led, 'R' for Red Led, 'B' for Blue Led and 'W' for White Led
GREEN
BLUE
BLUE
BLUE
BLUE
BLUE
WHITE
WHITE
WHITE
WHITE
RED
RED
RED
RED
RED
RED
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
HEY!!! I ONLY ACCEPT THE THREE LETTERS ABOVE!!!
```

**White led – serial communication**

Πάμε τώρα στις μετρήσεις μας. Οπως είπαμε και  νωρίτερα, επειδή η πλακέτα μας είναι χαμηλής ισχύος, δεν έχουμε μεγάλες διαφορές στην κατανάλωση.



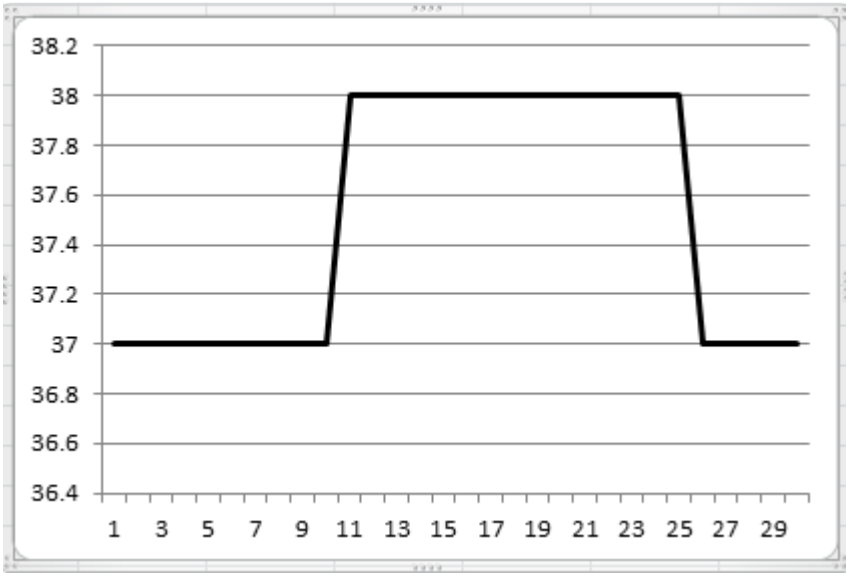

**Εικόνα 33:POWER CONSUMPTION OF BLUE-GREEN-RED-WHITE LED**

Στα πρώτα τρία δευτερόλεπτα όπου το πρόγραμμα μας δεν εκτελεί κάποια πράξη έχουμε μια κατανάλωση 37 mA. Πατώντας το πρώτο κουμπί ανάβει το μπλε φωτάκι (πρόσθεση) και έτσι έχουμε μια κατανάλωση 39 mA για τα επόμενα 7 δευτερόλεπτα. Με το που σβήσει το led η κατανάλωση ξανά πέφτει στα 37 mA για άλλα 3 δευτερόλεπτα.

Με το δεύτερο κουμπί(πολλαπλασιασμός) ανάβει το πράσινο led και καταναλώνει περίπου 42 mA για ακόμα 7 δευτερόλεπτα. Αφού σβήσει, πέφτει πάλι στα 37 mA για επιπλέον 3 sec. Το τρίτο κουμπί(διαίρεση) ενεργοποιεί το κόκκινο led και μας δίνει μια κατανάλωση στα 40 mA. Ομοίως και αυτό μόλις σβήσει θα πέσει η κατανάλωση στα 37 mA. Τέλος, πατώντας το τέταρτο κουμπί (σειριακή επικοινωνία χρήστη με υπολογιστή) ενεργοποιείται το λευκό led που έχει και τη μεγαλύτερη κατανάλωση στα 51 mA. Το πρόγραμμα αυτό το φτιάξαμε έτσι με τις καθυστερήσεις ανάμεσα στη κάθε πράξη έτσι ώστε να είναι όσο το δυνατό πιο εμφανή τα αποτελέσματα μας.

### 4.3.4 ΚΑΤΑΝΑΛΩΣΗ ΑΙΣΘΗΤΗΡΑ ΟΛΙΣΘΗΣΗΣ ΜΕ ΑΠΕΝΕΡΓΟΠΟΙΗΜΕΝΑ LEDS

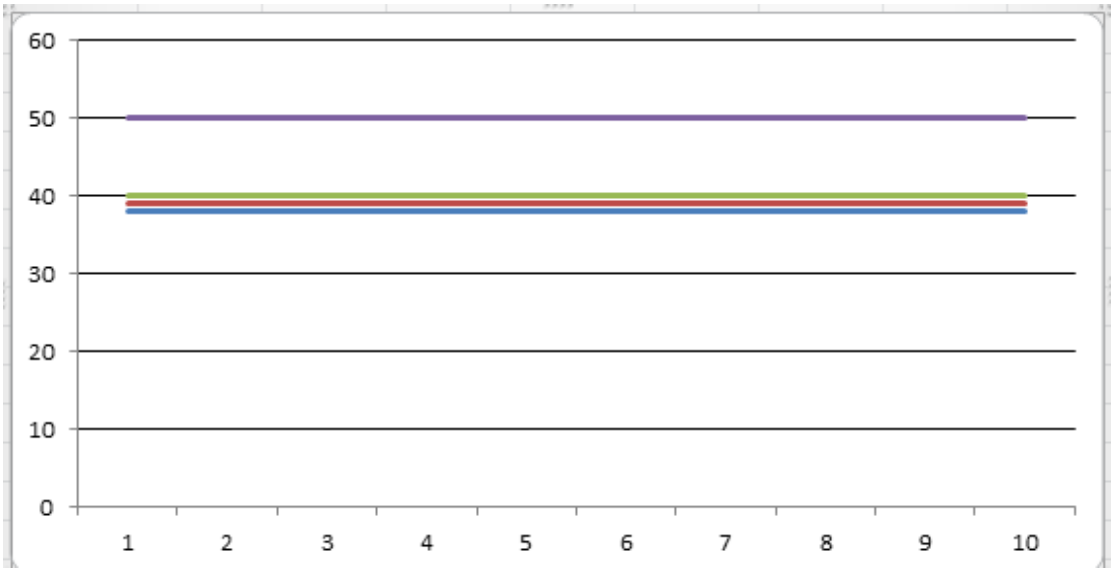
Είδαμε παραπάνω τις καταναλώσεις του αισθητήρα ολίσθησης με το κάθε led ξεχωριστά. Σ'αυτό το κομμάτι θα δούμε την κατανάλωση του slide bar με απενεργοποιημένα τα leds. Οπως φαίνεται στο παρακάτω γράφημα, στα πρώτα δέκα δευτερόλεπτα που δεν χρησιμοποιούμε τον αισθητήρα, έχουμε μια κατανάλωση της τάξης των 37 mA. Αμέσως μετά που πειράζουμε το slide bar πάνω κάτω για τα επόμενα δεκαπέντε δεύτερα, η κατανάλωση ανεβαίνει μόλις ένα mA, δηλαδή πάμε στα 38.Με το που το αφήνουμε, η κατανάλωση πέφτει πάλι στα 37 mA.




**Εικόνα 34:POWER CONSUMPTION OF SLIDE BAR WITHOUT LEDS**

### 4.3.5 ΚΑΤΑΝΑΛΩΣΗ LEDS

Ας δούμε τώρα πόσο καταναλώνει το κάθε led χωρίς να εκτελεί οτιδήποτε.Φαίνεται ξεκάθαρα στις εικόνες παρακάτω ότι, οι διαφορές στις καταναλώσεις είναι πάρα πολύ μικρές. Το μπλε led καταναλώνει 38 mA σε χρόνο 10 δευτερολέπτων. Το κόκκινο led καταναλώνει ένα mA παραπάνω, δηλαδή 39 mA. Το πράσινο led μας δίνει μια κατανάλωση στα 40 mA, ενώ η κατανάλωση του λευκού φτάνει και τα 50 mA.




**Εικόνα 35:POWER CONSUMPTION OF EACH LED**

# ΚΕΦΑΛΑΙΟ 5$^{Ο}$ – ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα πτυχιακή εργασία ξεκίνησε με μία παρουσίαση και μία ιστορική αναδρομή στα ενσωματωμένα συστήματα τα οποία κατακλύζουν το χώρο των ηλεκτρονικών σήμερα. Αφού μελετήθηκαν και περιγράφηκαν ενδελεχώς τα ενσωματωμένα συστήματα έγινε μία περιληπτική αναφορά των διαφόρων αρχιτεκτονικών δομών που χρησιμοποιούνται σε αυτά αλλά και των πλέον διασήμων λογισμικών με τα οποία γίνεται η ανάπτυξη και η παρουσίασή τους.

Στο πρακτικό της μέρος έγινε προσπάθεια αποτίμησης της απόδοσης αλλά και της κατανάλωσης των πολύ γνωστών επεξεργαστών ARM. Με στόχο την πολύ μικρή κατανάλωση αλλά και το ελάχιστο κόστος χρησιμοποιήθηκε η πλατφόρμα FRDM KL25Z της Freescale και το αμπερόμετρο Yocto για να αποφευχθεί οποιοσδήποτε κίνδυνος καταστροφής του υπολογιστή. Αφού μελετήθηκε η βασική αρχιτεκτονική των επεξεργαστών, οι ομάδες εντολών που μπορούν να χρησιμοποιηθούν αλλά και τα διάφορα περιφερειακούς έγινε επιλογή του κατάλληλου αναπτυξιακού περιβάλλοντος που θα παρείχε ευκολία στη χρήση αλλά και βελτιστοποίηση των δυνατοτήτων των παραπάνω χαρακτηριστικών.

Με χρήση διαφόρων προγραμμάτων / κωδίκων, έγινε καταγραφή πολλών μετρήσεων των μονάδων της απόδοσης/ισχύος υπό διαφορετικές συνθήκες λειτουργίας του επεξεργαστή, κάποιες από τις οποίες περιλάμβαναν και τη χρήση αισθητήρων. Στο σύνολο των μετρήσεων, παρατηρήθηκαν απειροελάχιστες διαφορές σε ότι αφορά την κατανάλωση του επεξεργαστή καθώς αυτός αποδεικνύεται να είναι ο πλέον ενεργειακά αποδοτικός.

Λαμβάνοντας υπ’ όψιν το σύνολο των μετρήσεων αλλά και τη συνολική απόδοση του επεξεργαστή διεξήχθη το συμπέρασμα ότι οι επεξεργαστές της οικογένειας ARM διαθέτουν χαρακτηριστικά όπως η απλότητα και η χαμηλή κατανάλωση που τους καθιστούν κατάλληλους για τη δημιουργία εφαρμογών χαμηλής ισχύος. Για το λόγο αυτό άλλωστε και έχουν υπερισχύσει σε αγορές τόσο κινητών τηλεφώνων όσο και ενσωματωμένων συστημάτων, ως οι πλέον κατάλληλοι, μικρού μεγέθους και χαμηλού κόστους μικροεπεξεργαστές.

## ΒΙΒΛΙΟΓΡΑΦΙΑ


*ARM*. (n.d.). Ανάκτηση από www.arm.com: http://www.arm.com/products/processors/cortex-m/cortex-m0plus.php

(2005). *ARM Architecture reference manual.*

(2014). *ARM White Paper.*

*Field Programmable Gate Arrays and Applications .*

*Getting Started With Keil.*

Greaves, D. (2011). System on a chip : Design and Modeling. University of Cambridge.

*IAR.* (n.d.). Ανάκτηση από www.iar.com: http://supp.iar.com/FilesPublic/UPDINFO/004916/arm/doc/EWARM_UserGuide.ENU.pdf

(2009). *IAR C/C++ Development .*

*Introduction to CodeWarrior.*

Katevenis, M. Risc Architectures. Στο *Parallel and Distributed Computing* (σσ. 595-619).

*Keil*. (n.d.). Ανάκτηση από www.keil.com

Masood, F. *Risc and Cisc Computer Architecture .*

*Wikipedia*. (n.d.). Ανάκτηση από https://en.wikipedia.org/wiki/System_on_a_chip

*Wikipedia*. (n.d.). Ανάκτηση από https://el.wikipedia.org/wiki/FPGA

*Wikipedia*. (n.d.). Ανάκτηση από https://en.wikipedia.org/wiki/Von_Neumann_architecture

*Wikipedia*. (n.d.). Ανάκτηση από https://el.wikipedia.org/wiki/%CE%91%CF%81%CF%87%CE%B9%CF%84%CE%B5%CE%BA%CF%84%CE%BF%CE%BD%CE%B9%CE%BA%CE%AE_%CE%A7%CE%AC%CF%81%CE%B2%CE%B1%CF%81%CE%BD%CF%84

*Wikipedia*. (n.d.). Ανάκτηση από https://en.wikipedia.org/wiki/CodeWarrior

Στο σημείο αυτό θα θέλαμε να ευχαριστήσουμε τους επιβλέποντες καθηγητές μας, κ.Βώρο Νικόλαο και τον κ.Αντωνόπουλο Χρηστο για την καθοριστική συμβολή τους και την πολύτιμη βοήθεια τους κατά τη διάρκεια εκπόνησης της πτυχιακής μας εργασίας.