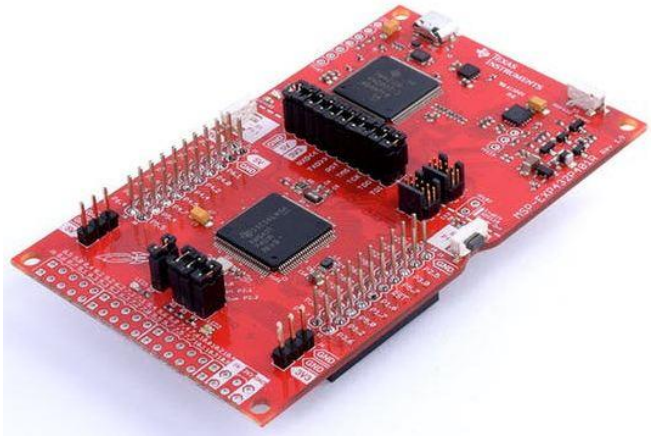
ΤΕΙ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: Ανάλυση Μικροελεγκτή τύπου ARM και υλοποίηση εφαρμογής ταξινόμησης υλικών σε ράφια

Επιβλέπων καθηγητής: Βισκαδούρος Γεώργιος

Φοιτητής: Καλογεράκης Ιωάννης

Αριθμός Μητρώου: ΤΗ5648

ΗΡΑΚΛΕΙΟ ΚΡΗΤΗΣ 2018

Ευχαριστίες

Θα ήθελα να ευχαριστήσω εκ βαθέων καρδίας τον κ. Βισκαδούρο Γεώργιο ο οποίος με εμπιστεύτηκε πως θα φέρω εις πέρας αυτή την πτυχιακή εργασία και ανέλαβε την επίβλεψη της. Καθώς για την στήριξη και τον πολύτιμό χρόνο του που μου αφιέρωσε για να φτάσουμε ως εδώ, όπως επίσης για την πολύτιμη καθοδήγηση και βοήθεια που μου πρόσφερε κατά την διάρκεια πραγματοποίησης της πτυχιακής μου εργασίας και τις πρακτικής μου άσκησης, καθώς για όλες εκείνες τις συμβουλές που με έκαναν να διαμορφώσω έναν σωστό τρόπο σκέψης για την επίλυση διάφορων προβλημάτων, δίνοντας μου έτσι σημαντικά εφόδια για την μετέπειτα ζωή μου.

Επίσης θα ήθελα να ευχαριστήσω θερμά τον κ. Συγγενίδη Χρήστο για την στήριξη, την βοήθεια και τις συμβουλές που μου πρόσφερε ανά πάσα ώρα και στιγμή κατά την διάρκεια της πρακτικής μου άσκησης, καθώς για την ευκαιρία που μου έδωσε να δω και να χρησιμοποιήσω καινοτόμα εργαλεία (Blynk app) για την επίτευξη της πτυχιακής μου εργασίας.

Τέλος θέλω να εκφράσω ένα τεράστιο ευχαριστώ στην μητέρα μου Κρυστάλλη Συντιχάκη και της αφιερώνω αυτή την πτυχιακή εργασία, για όλη αυτή την υποστήριξη, την συμπαράσταση, την εμπιστοσύνη που μου έδειξε όλα αυτά τα χρόνια των σπουδών μου και για όλα εκείνα τα εφόδια που μου πρόσφερε για να με κάνει έναν σωστό και ισορροπημένο άνθρωπο.

Περίληψη

Σκοπός της παρούσας πτυχιακής εργασίας, πρώτον είναι να δημιουργηθεί ένας οδηγός χρήσης ενός καινοτόμου μικροελεγκτή τύπου ARM για μελλοντική του χρήση στην έρευνα και σε μεταπτυχιακά προγράμματα. Έγινε λοιπόν η επιλογή του μικροελεγκτή της Texas Instruments MSP432P401R και του αναπτυξιακού MSP-EXP432P401R για τον λόγο της εξαιρετικά χαμηλής κατανάλωσης ισχύος και ταυτόχρονα της υψηλής απόδοσης του.

Και δεύτερον είναι να δημιουργηθεί μια εφαρμογή που θα τρέχει στον MSP-EXP432P401R και θα μας δίνει την δυνατότητα ταξινόμησης υλικών σε ράφια. Επίσης έγινε η κατασκευή μιας έξυπνης ραφιέρας που τρέχει την εφαρμογή μας και ελέγχεται μέσω κινητού τηλεφώνου.

Θα γίνει μια μικρή ανασκόπηση στα κεφάλαια που ακολουθούν.

- Στο πρώτο κεφάλαιο θα γίνει μια ιστορική αναδρομή στους μικροελεγκτές.
- Στο δεύτερο κεφάλαιο θα αναλυθεί η αρχιτεκτονική του MSP432P401R.
- Στο τρίτο κεφάλαιο γίνεται η περιγραφή του υλικού (Hardware).
- Στο τέταρτο κεφάλαιο βλέπουμε διάφορες εφαρμογές του μικροελεγκτή και την σύγκριση του με άλλους μικροελεγκτές.
- Το πέμπτο κεφάλαιο είναι ο οδηγός χρήσης του Code Composer Studio IDE που είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης του μικροελεγκτή μας από την ίδια την Texas Instruments, καθώς γίνεται και η ανάλυση στους βασικούς καταχωρητές του MSP432P401R.
- Το έκτο κεφάλαιο είναι ο οδηγός χρήσης του Energia IDE και των βασικών εντολών του, συνδέει την δομή (framework) του Arduino με τους μικροελεγκτές της Texas Instruments. Υπάρχει και τρίτος IDE με την ονομασία IAR Embedded Workbench αλλά δεν αναλύθηκε σε αυτή την πτυχιακή εργασία για τον λόγο ότι είναι επί πληρωμή σε σχέση με τους άλλους δύο που είναι ελεύθεροι για χρήση.
- Τέλος στο έβδομο κεφάλαιο έγινε η υλοποίηση εφαρμογής ταξινόμησης υλικών σε ράφια και η κατασκευή της έξυπνης ραφιέρας.

Abstract

The purpose of this thesis, first is to create a user guide of an innovative ARM microcontroller for future use in research and postgraduate programs. So the choice of the Texas Instruments MSP432P401R microcontroller and the MSP-EXP432P401R development board was made for the reason of extremely low power consumption and at the same time its high performance.

And secondly is to create an application that will run on the

MSP-EXP432P401R and will allow us to sort materials on shelves. We also built an intelligent rack that runs our application and is controlled through mobile phone.

It will be done a small review for the following chapters.

- In the first chapter there will be a historical review of microcontrollers.
- In the second chapter will be analyze the architecture of MSP432P401R.
- In the third chapter the description of the hardware is made.
- In the fourth chapter we see various applications of the microcontroller and its comparison with other microcontrollers.
- The fifth chapter is Code Composer Studio IDE user guide, which is the official integrated development environment of our microcontroller by Texas Instruments itself, as well as the analysis of basic registers of MSP432P401R.
- The sixth chapter is Energia IDE user guide and its basic commands, connects Arduino's structure to Texas Instruments microcontrollers. There is also a third IDE called IAR Embedded Workbench, but it wasn't analyzed in this thesis because it's for a fee in relation to the other two that are free to use.
- Finally in the seventh chapter was made the implementation of application of classification of materials on shelves and construction of smart rack.

Περιεχόμενα

1ο Κεφάλαιο. Ιστορική αναδρομή των μικροελεγκτών

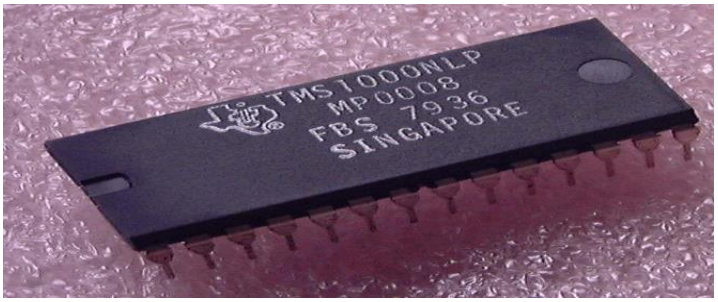
Γνωρίζεται ότι στις μέρες μας, ο καθένας από εμάς χρησιμοποιεί τουλάχιστον 20 μικροελεγκτές στο σπίτι του; Είναι γεγονός. Κάθε χρόνο παράγονται περισσότεροι από 2 δισεκατομμύρια μικροελεγκτές. Οι μικροελεγκτές χρησιμοποιούνται στις περισσότερες ηλεκτρονικές συσκευές για τις καθημερινές εφαρμογές του ανθρώπου, από τα απλά πλυντήρια έως το σύστημα αντιμπλοκαρίσματος φρένων στα αυτοκίνητα
[anti-lock brakes system (ABS)].

Η εφεύρεση τον μικροελεγκτών ξεκινάει κατά την διάρκεια του **1970** και **1971** όταν παράλληλα η Intel κατασκεύαζε τον πρώτο εμπορικό μικροεπεξεργαστή [Intel 4004 (4-bit)], ο Gary Boone τις Texas Instruments (TI) δούλευε σε μία παρόμοια ιδέα και κατασκεύασε τον πρώτο μικροελεγκτή με το όνομα TMS1802NC , ο οποίος ήταν ένα ενιαίο τσιπ ολοκληρωμένου κυκλώματος (integrated circuit chip) που περιείχε όλα εκείνα τα κυκλώματα που χρειάζονταν για να δημιουργήσει μια αριθμομηχανή (Calculator), με εξαίρεση την οθόνη και το πληκτρολόγιο τις αριθμομηχανής τα οποία τα έβαλε επιπρόσθετα.

Εικόνα 1. TMS 1802 NC - Πρώτος Μικροελεγκτής

Ωστόσο δεν ήταν μία συνηθισμένη συσκευή διότι αποτελούνταν από 5000 τρανζίστορς και πρόσφερε 3000 bits μνήμη μόνο ανάγνωσης [Read Only Memory (ROM)], δηλαδή μνήμη προγράμματος (Program Memory) και 128 bits μνήμη τυχαίας προσπέλασης [Random Access Memory (RAM)].

Το **1974** η Texas Instruments παρουσίασε την σειρά μικροελεγκτών TMS1000 One-Chip Microcomputers, ο οποίος συνδύαζε 4-bit κεντρικό επεξεργαστή 100 KHz έως 400 KHz, ROM , RAM, εισόδους / εξόδους [Inputs / Outputs (I/O)], το προϊόν έβγαινε στην αγορά σε διάφορες παραλλαγές στην RAM , ROM, χωρίς ROM (ROMless), με χρήση εξωτερικής ROM (External ROM), I/O, η γνήσια έκδοση (original version) έβγαινε με 1024 x 8 bits της ROM, 64 x 4 bits της RAM, και 23 I/O γραμμές και τεχνολογίας pMOS, στην συνέχεια η TI τους κυκλοφόρησε σε nMOS και CMOS.



Εικόνα 2. TMS 1000

Μέχρι το 1983 είχαν πουληθεί περίπου 100 εκατομμύρια TMS 1000.

Το **1976** η Intel παρουσίασε τον πρώτο τις μικροελεγκτή που ήταν ο 8048 (8-bits μικροεπεξεργαστή) και χρησιμοποιήθηκε στα πληκτρολόγια υπολογιστών της IBM, σε τηλεχειριστήρια τηλεοράσεων, στην κονσόλα παιχνιδιών Magnavox Odyssey και σε άλλα ηλεκτρονικά είδη ευρείας κατανάλωσης.

Το **1978** εγκαταστάθηκε η Acorn Computers στο Κέιμπριτζ.

Το **1980** η Intel κυκλοφόρησε τον μικροελεγκτή 8051 (8-bits μικροεπεξεργαστή, 128 bytes RAM, 4K bytes ROM) ο οποίος είναι ένας από τους διασημότερους μικροελεγκτές έως και στις μέρες μας.

Το **1979** η Motorola έφερε στην αγορά τον μικροελεγκτή 6809 (8-bits μικροεπεξεργαστή) και το **1982** έως **1984** κυκλοφόρησε το GCE Vectrex η Milton Bradley το οποίο ήταν πλατφόρμα παιχνιδιών.

Το **1984** η Motorola έδωσε στην αγορά τον διάσημο έως και σήμερα μικροελεγκτή 68HC11(8-bits μικροεπεξεργαστή), έχει χρησιμοποιηθεί σε πολλές εφαρμογές όπως σε εφαρμογές αυτοκινήτων, barcode readers, σε προγραμματιστές καρτών-κλειδιών στα ξενοδοχεία, σε ερασιτεχνικές εφαρμογές στην ρομποτική και σε διάφορα ενσωματωμένα συστήματα.

Το **1984** η Toshiba ανέπτυξε από τις Electrical Erasable and Programmable Read Only Memory [EEPROM] τις Flash μνήμες.

Το **1985** η Acorn Computer ανέπτυξε των πρώτο εμπορικό Reduced Instruction Set Computer ή αλλιώς RISC. Αγοράστηκε το 49% της Acorn από την Olivetti για 12 εκατομμύρια ευρώ σημερινά χρήματα μετά από την χρεοκοπία τις.

Κατά την διάρκεια του **1990** χρησιμοποιήθηκαν στους μικροελεγκτές οι Flash μνήμες (για την αποθήκευση του κυρίως προγράμματος) στους μικροελεγκτές λόγο ευκολίας διαγραφής και επαναπρογραμματισμού τους. Γνωστές εταιρίες μικροελεγκτών τις χρησιμοποίησε, όπως η Atmel στους AVR, η Microchip στους PIC.

Το **1987** κυκλοφόρησε σε χαμηλό κόστος ο πρώτος Acorn's ARM (Advanced RISC Machine) processor.

Το **1990** η Texas Instruments ανέπτυξε τον καινοτόμο μικροελεγκτή MSP430 (16-bits μικροεπεξεργαστή), ο οποίος πήγαινε την ενσωματωμένη επεξεργασία στο επόμενο επίπεδο, χαμηλή κατανάλωση ισχύος και την αποδοτική εκμετάλλευση της. Καθώς παρουσιάζει των πρώτο επεξεργαστή για εφαρμογές στα κινητά τηλέφωνα (OMAP), την σειρά TMS320.

Το **1990** συνεργάστηκε η Acorn με την Apple για να αναπτύξουν ARM επεξεργαστή ως μέρος για το σχέδιο Newton Computer System της Apple.

Το **1996** η Atmel ανάπτυξε τους AVR (Alf and Vegard's RISC processor).

Το **2005** κυκλοφόρησε η πλατφόρμα του Arduino και χαρακτήριζε των ATmega8 AVR μικροελεγκτή.

Το **2011** ο Steve Ballmer, γενικός διευθυντής τις Microsoft είπε "η επόμενη γενιά των λειτουργικών συστημάτων Windows θα βασίζεται σε μικροτσίπ σχεδιασμένα σε ARM".

Το **2015** η Texas Instruments κυκλοφόρησε τον MSP432 μικτών σημάτων μικροελεγκτή βασισμένο στον 32-bit ARM Cortex-M4F CPU και επεκτείνει την σειρά των 16-bit MSP430, έχει μεγαλύτερο χώρο για διεύθυνσης μνήμης (έως 4GB) και κώδικα (32 bits), καθώς μεγαλύτερη επεξεργαστική ισχύς (έως 48 MHz), όπως ο προκάτοχος του MSP430 έτσι και ο MSP432 έχει κατασκευαστεί για να δέχεται πολλές περιφερειακές συσκευές και είναι σχεδιασμένος να έχει πολύ χαμηλή κατανάλωση ισχύος.

Το **2016** η Microchip Technology αγοράζει την Atmel.

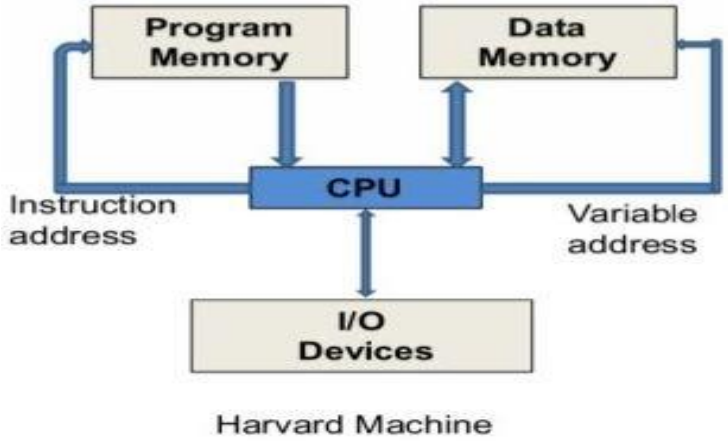
Τον Μάιο του **2018** ανακοινώθηκε ο Cortex-M35P core.

Οι μικροελεγκτές έχουν γίνει ένα αναπόσπαστο κομμάτι της καθημερινότητας μας και η ιστορία τους συνεχίζεται.

2ο Κεφάλαιο. Αρχιτεκτονική μικροελεγκτή MSP432P4xx

Αρχιτεκτονική Harvard

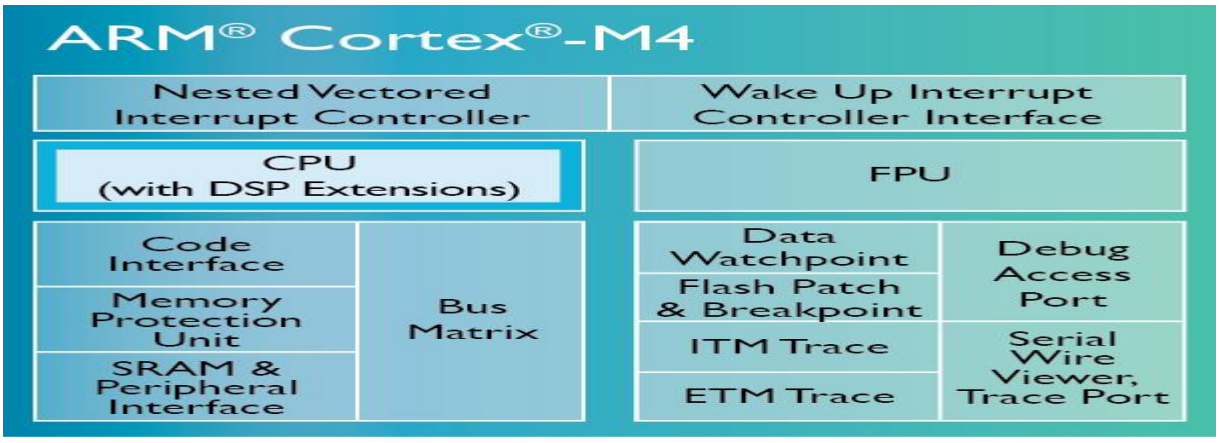
Ο μικροελεγκτής MSP432P4xx χρησιμοποίει τον επεξεργαστή (CPU) 32-bit ARM Cortex-M4F οπού χρησιμοποιεί την αρχιτεκτονική του Harvard, που σημαίνει ότι έχει ξεχωριστό δίαυλο και αποθηκευτικό χώρο εντολών (Instruction bus, Instruction memory) και ξεχωριστό δεδομένων (Data bus, Data memory), αυτό επιτυγχάνει μεγαλύτερες ταχύτητες εκτέλεσης του προγράμματος.



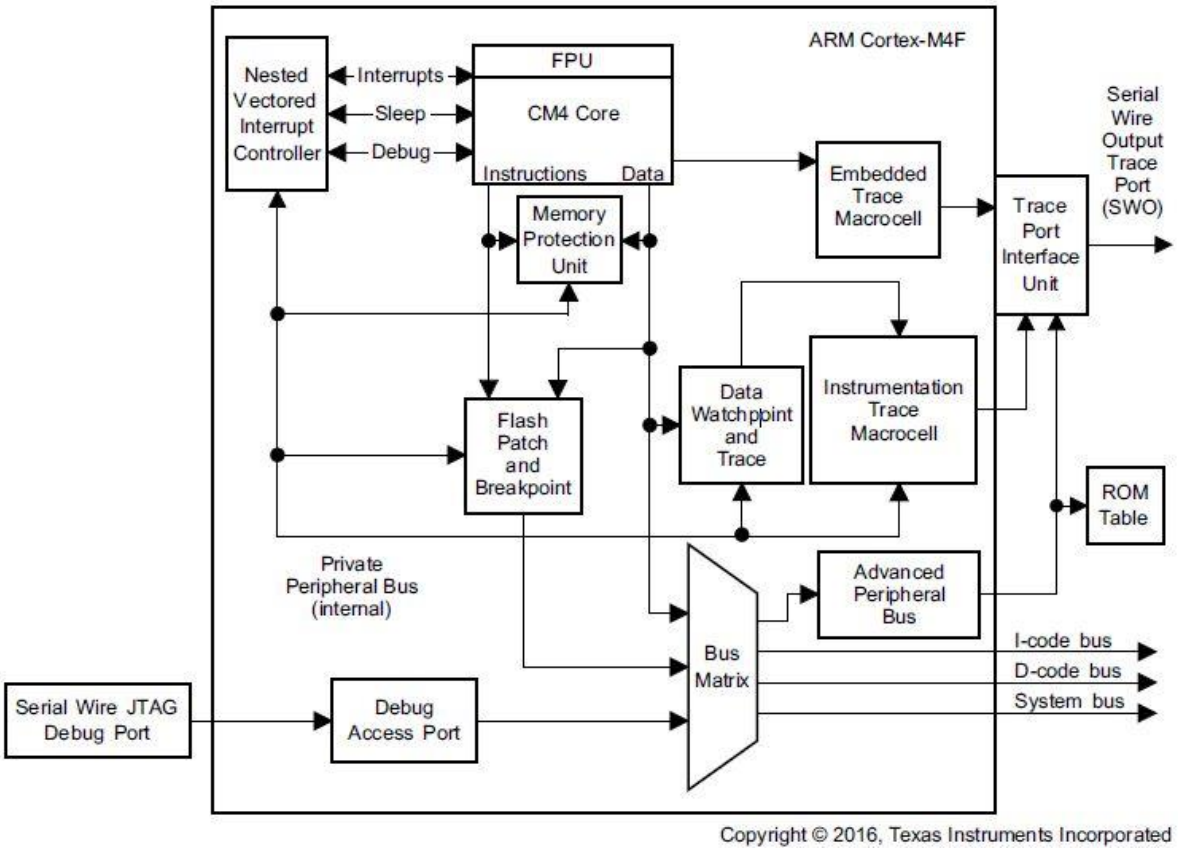
Εικόνα 3. Αρχιτεκτονική Harvard

Αρχιτεκτονική ARM Cortex-M4F

Αναλυτικότερα ο Cortex-M4F επεξεργαστής είναι κατασκευασμένος για υψηλές επιδόσεις, βασισμένος στην αρχιτεκτονική ARMV7-M, στην Εικόνα 4. βλέπουμε την αρχιτεκτονική του ARM Cortex-M4F και αναλύεται στο μπλοκ διάγραμμα της Εικόνας 5. που ακολουθεί παρακάτω.



Εικόνα 4. Αρχιτεκτονική ARM Cortex-M4F



Εικόνα 5. Μπλοκ διάγραμμα της Κεντρικής Μονάδας Επεξεργασίας (CPU Block Diagram) - Arm Cortex-M4F

Η εφαρμογή του Cortex-M4F στον MSP432P4xx όπως φαίνεται στο μπλοκ διάγραμμα της Εικόνας 5. ενσωματώνει τα εξής:

- Ο πυρήνας του επεξεργαστή είναι ο ARM Cortex-M4 processor core (Revision r0p1) βασισμένος στην αρχιτεκτονική ARMV7-M
- Έχει εμφωλευμένο ελεγκτή για τον έλεγχο τον διακοπών [Nested Vectored Interrupt Controller (NVIC)] κοντά στον πυρήνα του επεξεργαστή, για να επιτύχει χαμηλή καθυστέρηση στην επεξεργασία των διακοπών (Interrupts), καθώς και γρήγορη εκτέλεση των υπηρεσιών ρουτινών διακοπών [Interrupt Service Routines (ISRs)]
    - Υποστηρίζει 64 πηγές διακοπών
    - Τρία bits διευκρινίζουν την προτεραιότητα της κάθε διακοπής (interrupt) συνολικά οκτώ επίπεδα προτεραιότητας
    - Δυναμική ρύθμιση προτεραιότητας των Interrupts
    - Περιλαμβάνει αμασκάριστη διακοπή [nonmaskable interrupt (NMI)] για τις μη ανακτήσιμες καταστάσεις του υλικού (Hardware), για προβλήματα στην

απασφαλμάτωση του συστήματος και αναλαμβάνουν ειδικές υποθέσεις παράδειγμα να επανεκκινήσουν το σύστημα (System Reset)

- Έχει πολλαπλές διασυνδέσεις διαύλων υψηλής επίδοσης (Multiple high-performance bus interfaces)
- Χαμηλού κόστους λύσεις απασφαλμάτωσης (Debug) με τις ιδιότητες:
    - Εφαρμογή σημείων διακοπής (Breakpoints) μέσω του Flash Patch Breakpoint Unit (FPB)
    - Εφαρμογή σημείων παρακολούθησης (Watchpoints), μέσω ιχνηλασίας (Tracing) και την διαμόρφωση του συστήματος μέσω του Data Watchpoint and Trace (DWT)
    - Υποστήριξη για θύρες απασφαλμάτωσης που βασίζονται σε σειριακό καλώδιο (Serial Wire) και το JTAG (ονομασία από τα αρχικά Joint Test Action Group)
    - Πρόσβαση απασφαλμάτωσης σε όλη την μνήμη και τους καταχωρητές του συστήματος, συμπεριλαμβανομένης πρόσβασης στις συσκευές που έχουν χαρτογραφηθεί στην μνήμη (memory mapped devices), πρόσβαση στους εσωτερικούς καταχωρητές του πυρήνα όταν αυτός είναι σταματημένος και πρόσβαση στου καταχωρητές απασφαλμάτωσης ακόμη και όταν το SYSRESETn (Το σύστημα σε επανεκκίνηση) είναι σε ισχύς
    - Παρακολούθηση της μακροκυψέλης [Instrumentation Trace Macrocell (ITM)] για την υποστήριξη του στυλ απασφαλμάτωσης printf()
    - Μονάδα διεπαφής παρακολούθησης θύρας [Trace Port Interface Unit (TPIU)] για την γεφύρωση του αναλυτή παρακολούθησης θύρας [Trace Port Analyzer (TPA)], συμπεριλαμβανομένου της λειτουργίας μονής εξόδου [Single Wire Output (SWO) mode]
- Μονάδα προστασίας μνήμης [Memory Protection Unit (MPU)], υποστηρίζει οχτώ περιοχές (Regions), καθώς έχει και απενεργοποιημένες υπό περιοχές μνήμης [Sub Region Disable (SRD)] που τις ενεργοποιεί αποδοτικά
- Διαθέτει την μονάδα IEEE 754-compliant κινητής υποδιαστολής [Floating Point Unit (FPU)] που του επιτρέπει γρήγορη επεξεργασία των μαθηματικών πράξεων κινητής υποδιαστολής, 32-bit εντολές (Instructions) για μονής ακριβείας (C Float) επεξεργασίας δεδομένων (Data processing operation)

- Υποστηρίζει Bit-banding [Χαρτογράφηση των bits μιας ολόκληρης περιοχής μνήμης σε μια ολόκληρη λέξη (entire word) στην άλλη περιοχή Bit-band Alias Region] για την στατική μνήμη τυχαίας προσπέλασης [Static Random Access Memory (SRAM)]
- Διαθέτει SysTICK χρονιστή (timer) για περιοδικά τικ (ticks), είναι 24 bits μετρητής προς τα κάτω (Count Down Timer), βοηθάει το σύστημα για πολλαπλές διεργασίες (multitasking) κρατώντας, μετρώντας τον χρόνο, καθώς και σαν λειτουργεί πηγή διακοπών για βασικές λειτουργίες, μπορεί να χρησιμοποιηθεί για χρονιστής σε λειτουργικό πραγματικού χρόνου [Real-Time Operating System (RTOS)] ή σαν απλός μετρητής
- Διαύλους διεπαφής (Bus Interfaces):
    - Τρείς ανώτερους διαύλους υψηλής επίδοσης [Three Advanced High-performance Bus-Lite (AHB-Lite)], διεπαφές(interfaces):
        - ICode bus χρησιμοποιείται για την πρόσβαση εντολών (Instructions) εντός του εύρους διευθύνσεων, συνδέεται με την Flash μνήμη, με την ROM και την SRAM.
        - DCode bus χρησιμοποιείται για την πρόσβαση δεδομένων (Data) εντός του εύρους διευθύνσεων, συνδέεται με την Flash μνήμη, με την ROM και την SRAM.
        - System bus χρησιμοποιείται για την πρόσβαση δεδομένων εντός του εύρους διευθύνσεων, συνδέεται με την SRAM όχι όμως για την εκτέλεση κώδικα παρά μόνο την αποθήκευση δεδομένων και της στοίβας, καθώς για τα πάνω στο τσιπ περιφερειακά (on-chip peripherals)
    - Προσωπικό δίαυλο περιφερειακών [Private Peripheral Bus (PPB)] βασισμένο στην προηγμένη διεπαφή περιφερειακού δίαυλου [Advanced Peripheral Bus (APB) interface]
    - Πρόσβαση ευθυγράμμισης μνήμης (Memory access alignment)

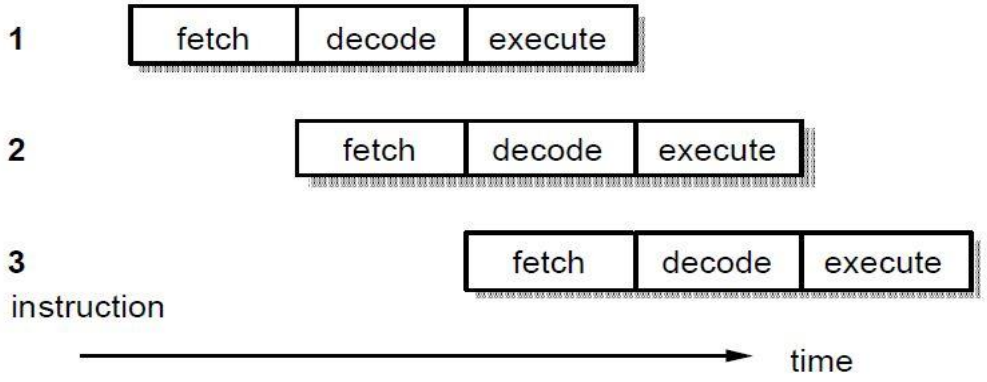
Ο επεξεργαστής υποστηρίζει ψηφιακή επεξεργασία σημάτων [Digital Signal Processing (DSP)], έχει έλεγχο ψηφιακών σημάτων [Digital Signal Control (DSC)], καθώς υποστηρίζει πολλαπλά δεδομένα μονής οδηγίας [Single Instruction Multiple Data (SIMD)] και πρότυπο μονής ακρίβειας (single precision) κινητής υποδιαστολής [Floating-point unit (FPU)] που του δίνει το χαρακτηριστικό γράμμα "F" στο όνομα του, αλλιώς αν δεν είχε το FPU θα ήταν σκέτο Cortex-M4.

3-Stage Pipeline

Ο επεξεργαστής λειτουργεί με την τεχνική 3-stage pipeline του Harvard με την μελέτη σε κλάδους (branch speculation) που τον καθιστά ιδανική επιλογή για τις ενσωματωμένες εφαρμογές (embedded applications), καθώς αυξάνει την ταχύτητα εκτέλεσης τον προγραμμάτων επίσης δίνει την δυνατότητα πολλαπλής εκτέλεσης προγραμμάτων (Multitasking).

Τα 3 στάδια του pipeline είναι τα εξής:

- Πρώτο στάδιο, Φέρω (Fetch) : Φέρνει την εντολή από την μνήμη
- Δεύτερα στάδιο, Αποκωδικοποίηση (Decode) : Η εντολή αποκωδικοποιείτε και η διαδρομή δεδομένων (Datapath) για τα σήματα ελέγχου προετοιμάζεται για τον επόμενο κύκλο
- Τρίτο στάδιο, Εκτέλεση (Execute) : οι τελεστές (Operands) διαβάζονται από των καταχωρητή τράπεζας (Register Bank) που βρίσκεται στην FPU, μετατοπίζεται (Shifted), συνδυάζεται στην Αριθμητική Λογική Μονάδα [Arithmetic Logic Unit (ALU)] και επιστρέφει το αποτέλεσμα



Εικόνα 6. Τα 3 στάδια του pipeline

Υποστηρίζει εντολές ενός κύκλου (Single cycle instructions), δηλαδή ολοκληρώνονται με ρυθμό ανά ένα χτύπο ρολογιού (one per clock cycle).

Όπως φαίνεται στην Εικόνα 6. σε έναν χτύπο ρολογιού εκτελούνται 3 εντολές, φέρνει την πρώτη εντολή από την μνήμη και ξεκινάει να την αποκωδικοποιεί, στον χρόνο που αποκωδικοποιεί την πρώτη φέρνει την δεύτερη εντολή, έπειτα εκτελεί την πρώτη εντολή και ξεκινάει την αποκωδικοποιήσει της δεύτερης και ταυτόχρονα φέρνει την τρίτη εντολή στον ίδιο χρόνο, στην συνέχεια εκτελεί την δεύτερη εντολή και αποκωδικοποιεί την τρίτη, τέλος

εκτελεί και την τρίτη εντολή, ουσιαστικά εκμεταλλεύεται τα κενά χρόνου (Time Slots) που δημιουργούνται μεταξύ των 3 σταδίων.

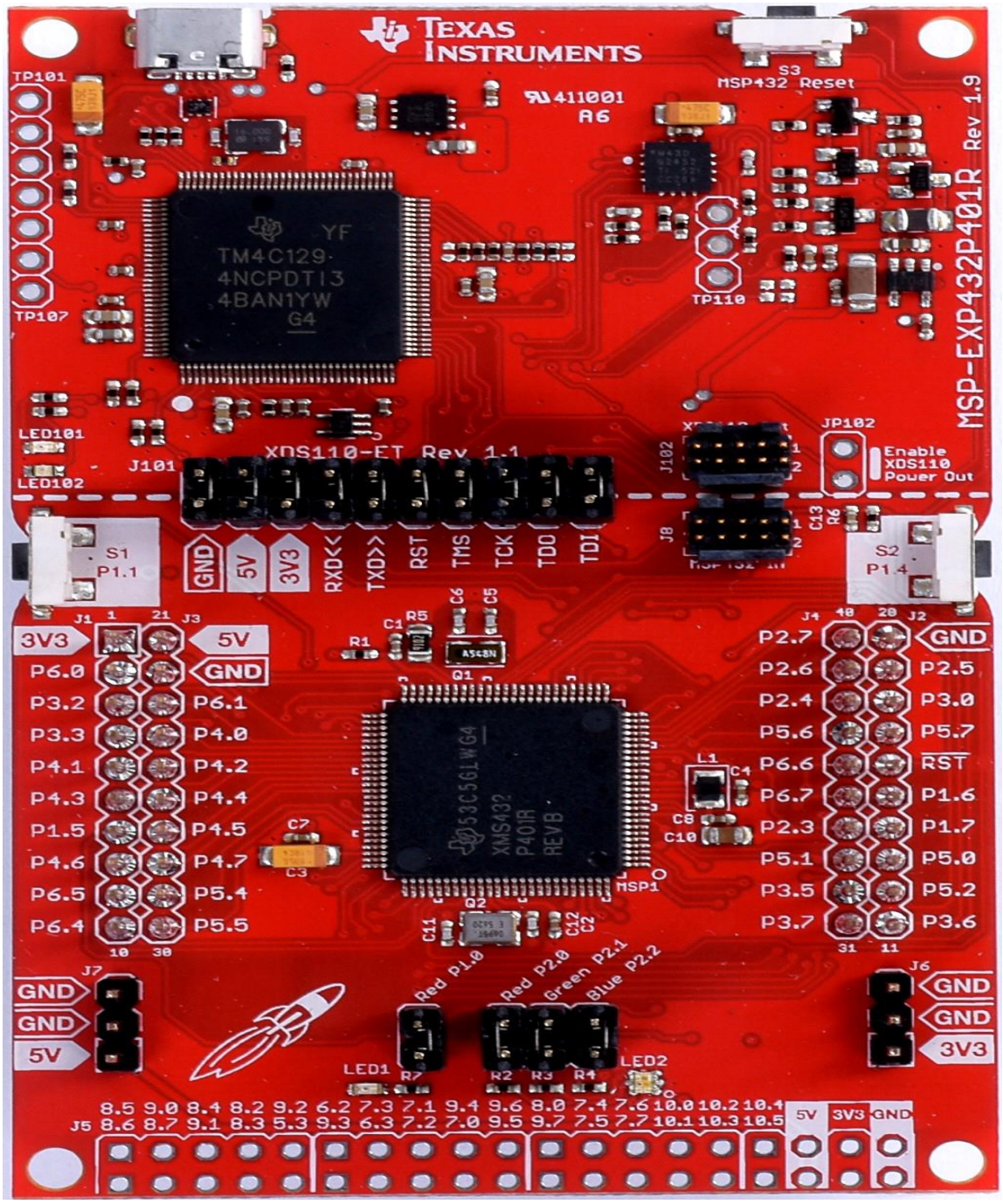
Ο μετρητή προγράμματος [Program Counter (PC)] πού είναι ο καταχωρητής R15 συμπεριφέρεται ως εξής:

- αυξάνει 2 φορές πριν εκτελεστή η εντολή, λόγω της λειτουργίας του pipeline
- επομένως ο R15 έχει την διεύθυνση εντολής + 8 (+12 εάν χρησιμοποιηθεί μετά τον πρώτο κύκλο εάν και αυτό είναι αρχιτεκτονικά απροσδιόριστο), στον Thumb code το offset είναι +4
- κανονικά ο συμβολομεταφράστης (assembler) κάνει της αναγκαίες προσαρμογές π.χ. κλάδους (Branches)

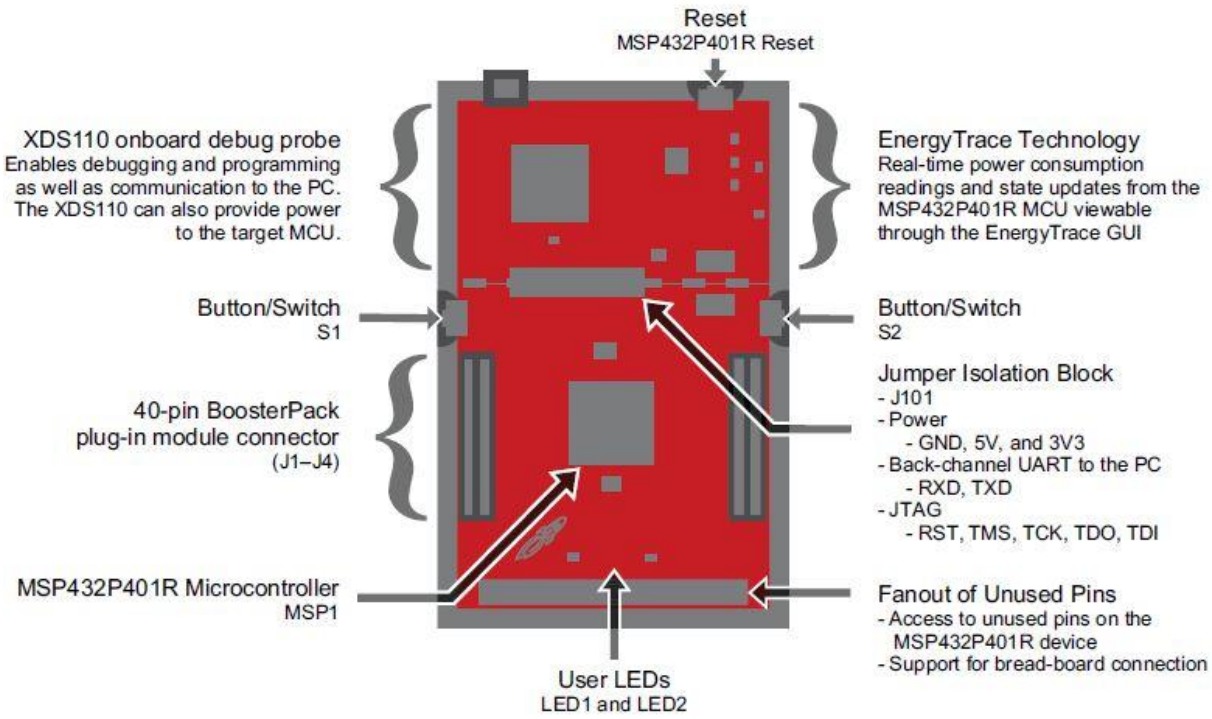
3ο Κεφάλαιο. Περιγραφή του υλικού (Hardware description)

Αναπτυξιακό (Development Kit) MSP-EXP432P401R®. LaunchPad

Στην συγκεκριμένη πτυχιακή εργασία χρησιμοποιήθηκε το αναπτυξιακό MSP-EXP432P401R της Texas Instruments®.

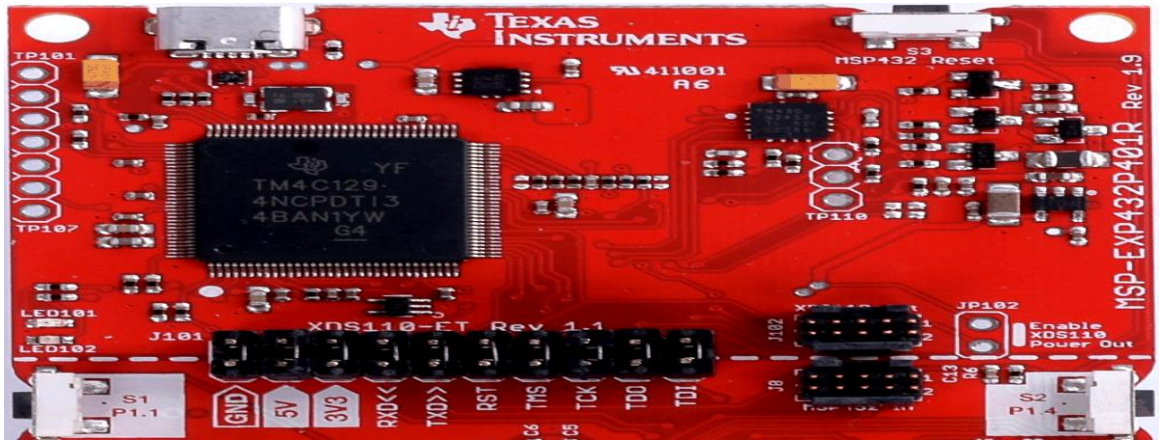


Εικόνα 7. Αναπτυξιακό MSP-EXP432P401R®.



Εικόνα 8. Περιγραφή του αναπτυξιακού MSP-EXP432P401R®.

Όπως βλέπουμε στην Εικόνα 7 και 8. πάνω αριστερά του αναπτυξιακού είναι το XDS110 το οποίο είναι ενσωματωμένος (onboard) ακροδέκτης απασφαλμάτωσης (Debug Probe) που συνδέει τον μικροελεγκτή με τον υπολογιστή δίνοντας του την δυνατότητα προγραμματισμού και απασφαλμάτωσης του, καθώς είναι και το βύσμα τροφοδοσίας του, είναι microUSB. Ακολουθεί η Εικόνα 9. που είναι το κομμάτι της πλακέτας του XDS110 και του EnergyTrace.



Εικόνα 9. XDS110-ET debug probe

Στην συνέχεια κάτω από το XDS110 είναι το εσωτερικό κουμπί / διακόπτης (Button / Switch) με την ονομασία S1 το οποίο μπορούμε να το προγραμματίσουμε κανονικά.

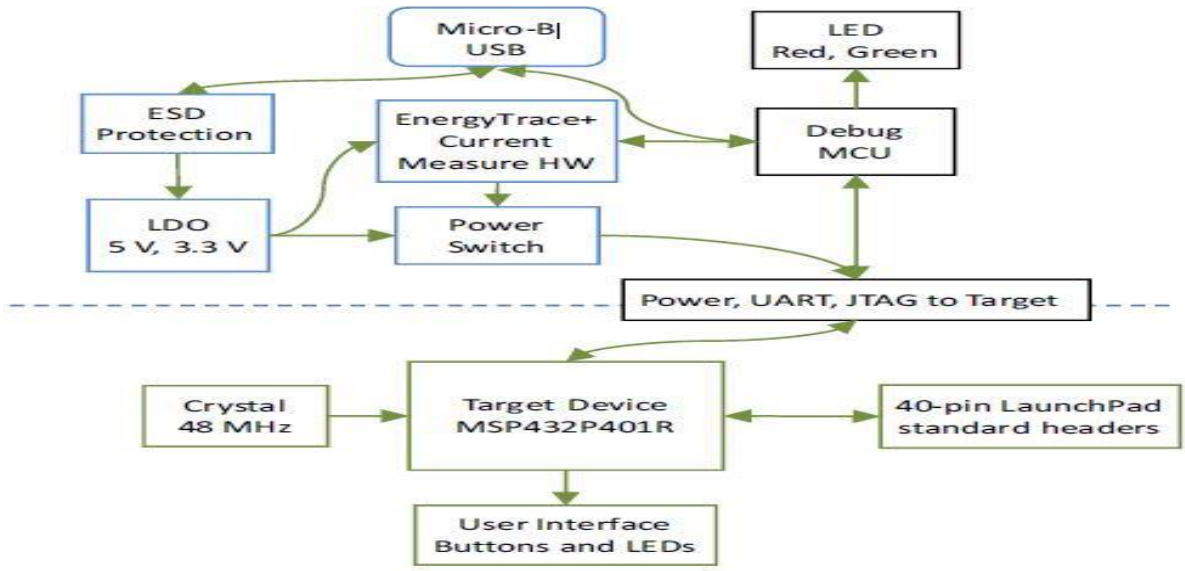
Ακολουθεί κάτω από το κουμπί S1 και δεξιά αριστερά του αναπτυξιακού συνολικά 40 ακροδέκτες σύνδεσης (Pins) πού είναι οι θύρες συνδέσεων του μικροελεγκτή από J1 έως J4 φαίνονται ξεκάθαρα στην Εικόνα 7.

Πάνω δεξιά του αναπτυξιακού βλέπουμε το κουμπί της επανεκκίνησης (Reset Button).

Κάτω από το κουμπί του Reset βλέπουμε την τεχνολογία ανίχνευσης ενέργειας (EnergyTrace Technology) η οποία μας δείχνει σε πραγματικό χρόνο την κατανάλωση ενέργειας του μικροελεγκτή (MSP432P401R MCU) και μέσω του γραφικού περιβάλλοντος [Graphical User Interface (GUI)] EnergyTrace GUI μπορούμε να την δούμε σαν γραφική παράσταση (υποστηρίζεται από το πρόγραμμα Code Composer Studio στο οποίο θα γίνει ανάλυση του στο 5^{ο} κεφάλαιο).

Ακριβώς κάτω από το EnergyTrace είναι το κουμπί με την ονομασία S2.

Ενδιάμεσα των κουμπιών S1 και S2 είναι το μπλοκ γεφυρώσεων απομόνωσης (Jumper Isolation Block) με την ονομασία J101, ουσιαστικά επιτρέπει στον χρήστη να συνδέει ή να αποσυνδέει τα σήματα που διέρχονται από τον τομέα τις πλακέτας του XDS110-ET της Εικόνας 9. με τον τομέα του MSP432P401R η διαχωριστική γραμμή των 2 τομέων απεικονίζεται σαν μια λευκή διακεκομμένη γραμμή πάνω στο αναπτυξιακό, φαίνεται ξεκάθαρα στην Εικόνα 7,9 και 10.



Εικόνα 10. Μπλοκ διάγραμμα του αναπτυξιακού MSP-EXP432P401R®

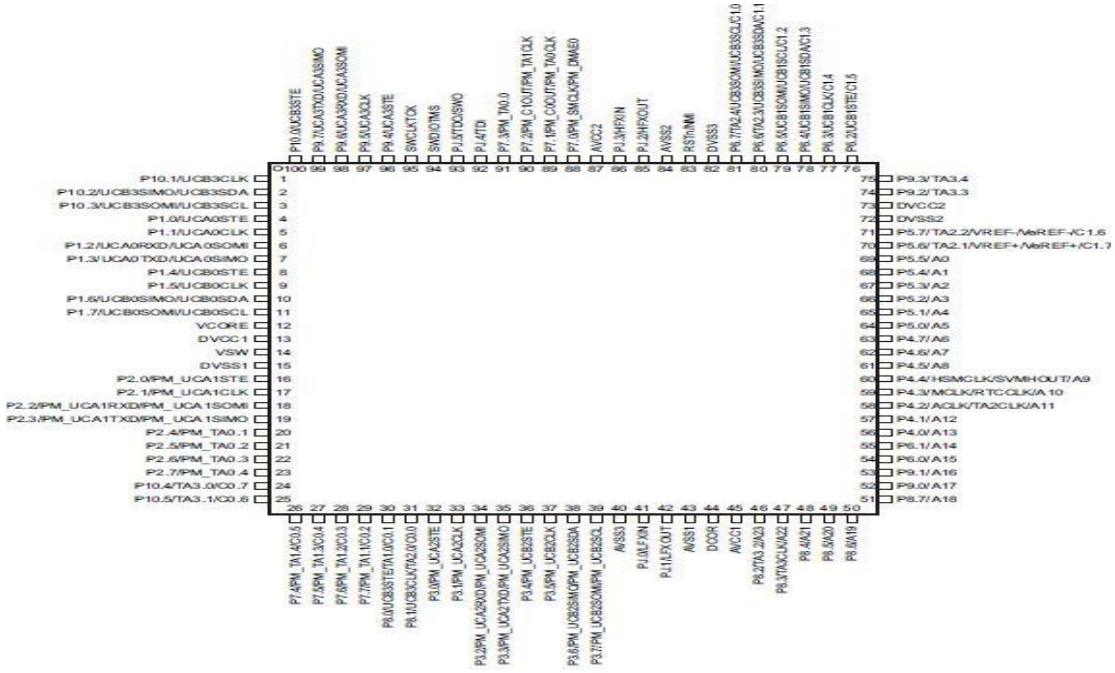
Το XDS110-ET μπορεί να αποσυνδεθεί τελείως από την μεριά του MSP432P401R διότι όλα τα σήματα διέρχονται μέσο του J101 block, συμπεριλαμβανομένου της γείωσης GND, UART και του JTAG.

Στην συνέχεια κάτω από το block γεφυρώσεων J101 και ενδιάμεσα στις θύρες του μικροελεγκτή βλέπουμε το τσιπ του μικροελεγκτή MSP432P401R. Ακολουθεί η Εικόνα 11. που είναι το τσιπ του MSP432P401R.

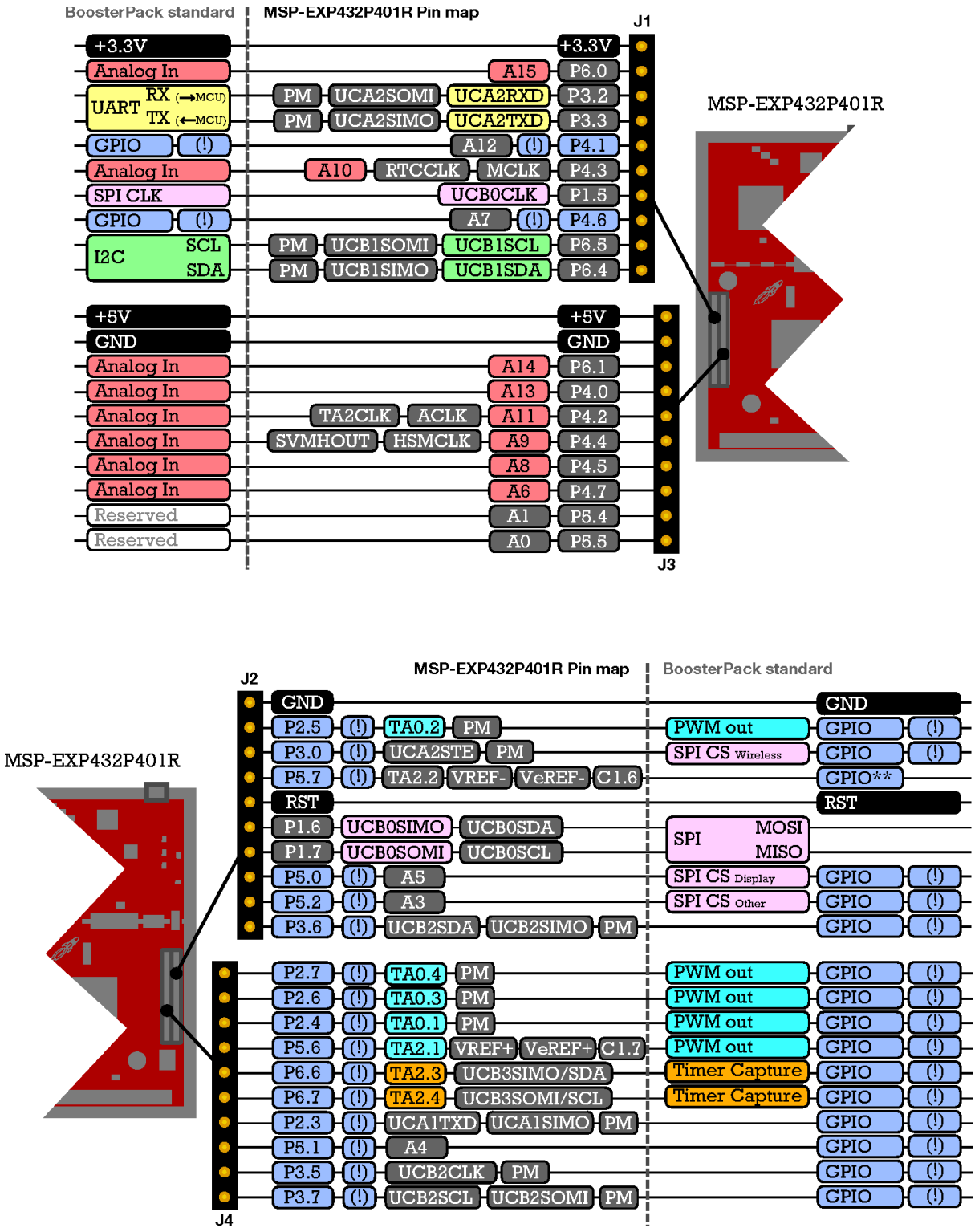
Εικόνα 11. Τσιπ μικροελεγκτή MSP432P401R®

Κάτω από το τσιπ του μικροελεγκτή είναι το LED1 το οποίο είναι κόκκινο χρώμα και το LED2 που είναι τριών χρωμάτων κόκκινο, πράσινο, μπλε [Red, Green, Blue (RGB)].

Τέλος κάτω από τα LED1 και LED2 είναι οι υπόλοιπες θύρες του μικροελεγκτή οι οποίες δεν έχουν μεταλλικούς ακροδέκτες συνδεμένους επάνω.



Εικόνα 12. Αναλυτική λειτουργία των ποδιών / ακροδεκτών του μικροελεγκτή (MSP432P401RIPZ® Pinout)



Εικόνα 13. Χάρτης λειτουργιών για τους ακροδέκτες (Pin map) στις θύρες συνδέσεων J1 έως J4 για το πρότυπο τον 40-pin

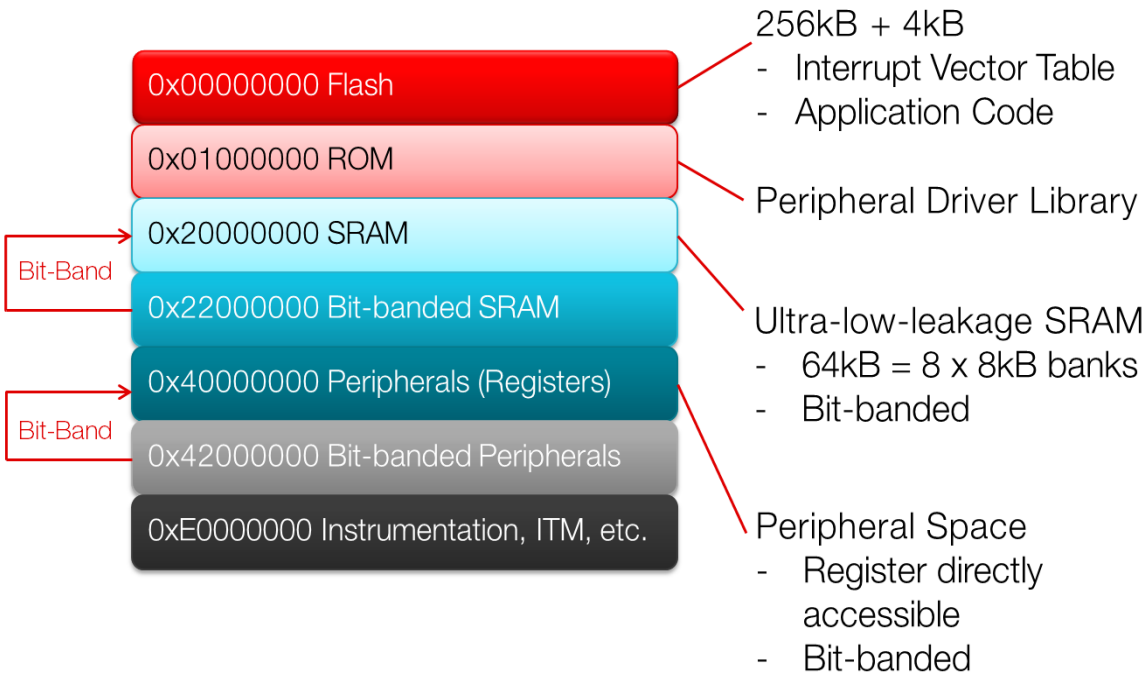
Στην Εικόνα 13. βλέπουμε αναλυτικά τις λειτουργίες του κάθε Pin που βρίσκεται στις θύρες συνδέσεων J1 έως J4 ή σε νούμερα από 1 έως 40 pin.

(!) Το θαυμαστικό σημαίνει ότι η αντίστοιχη I/O μπορεί να υποστηρίξει Interrupt.

Τεχνικά χαρακτηριστικά του μικροελεγκτή MSP432P401R®

Ο MSP432P401R είναι η πρώτη οικογένεια συσκευών MSP432 που διαθέτει χαμηλή κατανάλωση ισχύος και υψηλής απόδοσης πυρήνα Arm Cortex-M4F. Στα τεχνικά χαρακτηριστικά της συσκευής συμπεριλαμβάνονται τα εξής:

- Χαμηλής κατανάλωσης 32-bit Arm Cortex-M4F MSP432P401R, ενεργός με λιγότερα από 100 μA / MHz, διατήρηση (Retention) μνήμης RAM με το ρολόι πραγματικού χρόνου [Real Time Clock (RTC)] με κατανάλωση λιγότερο από 1 μA, δυνατότητες λειτουργίας χαμηλής κατανάλωσης (Low Power Mode), ύπνου (Sleep Mode) και βαθύ ύπνου (Deep Sleep Mode)
- Ρολόι συστήματος (system clock) έως 48 MHz
- 256 KB flash memory, 64 KB σε SRAM και 32 KB σε ROM μαζί με τις βιβλιοθήκες του κιτ ανάπτυξης λογισμικού [Software Development Kit (SDK)] SimpleLink MSP432, στην Εικόνα 13. βλέπουμε τον χάρτη μνήμης του MSP432P401R



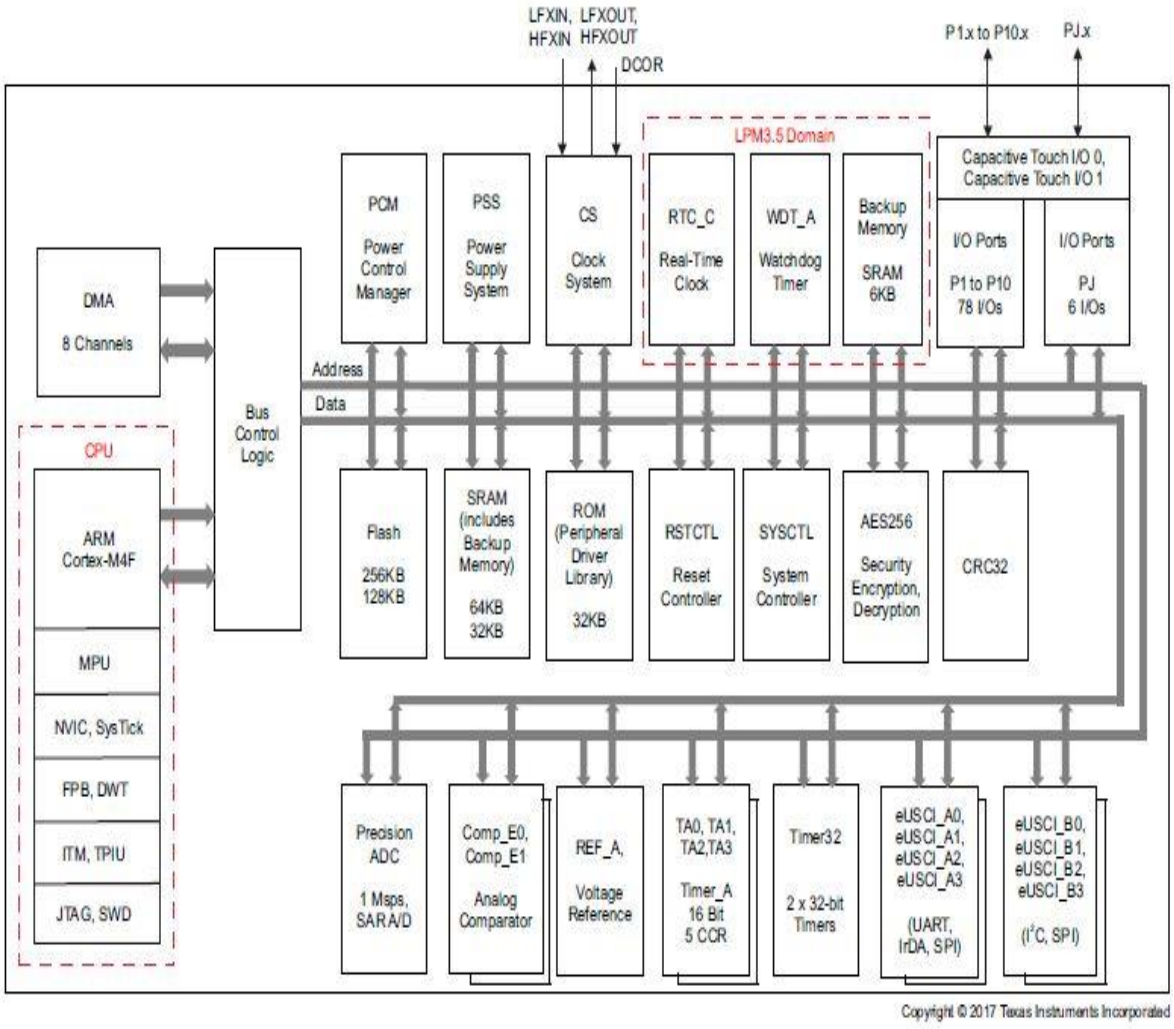
Εικόνα 14. Χάρτης μνήμης του μικροελεγκτή (MCU Memory Map) MSP432 (MSP432P401R)

- Τέσσερις 16-bit χρονιστές (timers) με λειτουργίες σύλληψης (capture), σύγκρισης (compare), ή Διαμόρφωση Πλάτος Παλμού [Pulse Width Modulation (PWM)], Δύο 32-bit timers και ένα ρολόι πραγματικού χρόνου (RTC)

- Έως 8 σειριακά κανάλια επικοινωνίας (serial communication channels) τα εξής: I^2C, SPI, UART και IrDA
- Αναλογικά (Analog): 14-bit ακριβείας (precision) και 1 Mega Samples Per second (MSPS) Μετατροπέα Αναλογικού σε Ψηφιακό [Analog to Digital Converter (ADC)], χωρητική επαφή (capacitive touch), συγκριτή (comparator)
- Ψηφιακά (Digital): AES256, Κυκλικό Έλεγχο Πλεονασμού [Cyclic Redundancy Check (CRC)], μDMA

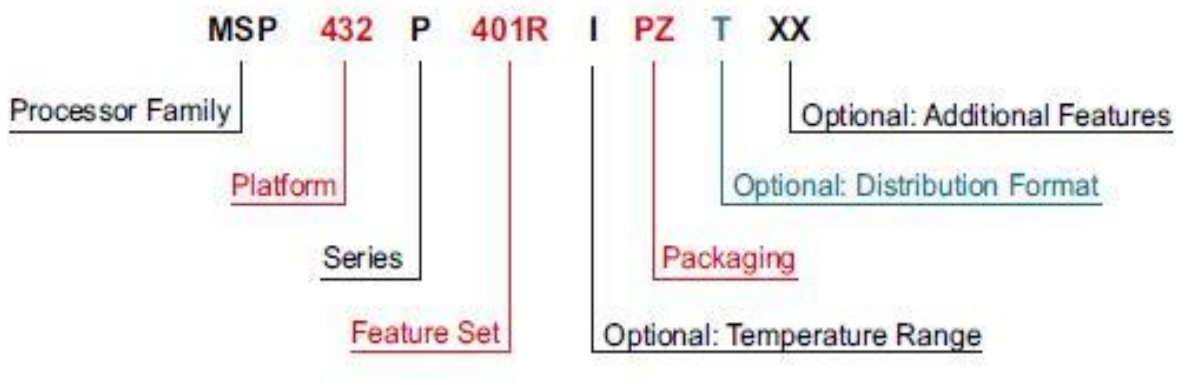
Για περαιτέρω ανάλυση των τεχνικών χαρακτηριστικών ακολουθεί ο ηλεκτρονικός σύνδεσμος (Link) της Texas Instruments: http://www.ti.com/lit/ds/symlink/msp432p401r.pdf

Γνωρίζοντας τα τεχνικά χαρακτηριστικά και τις συσκευές που υπάρχουν πάνω στο αναπτυξιακό μπορούμε να δούμε το λειτουργικό μπλοκ διάγραμμα (functional block diagram) συσκευών του MSP432P401R (είναι και για τον MSP432P401M που δεν αναλύεται σε αυτή την πτυχιακή εργασία).



Εικόνα 15. Functional Block Diagram συσκευών του MSP432P401R και MSP432P401M

Ονοματολογία συσκευών (Device Nomenclature)



Εικόνα 16. Ανάλυση ονομάτων

Πίνακας 1. Ονοματολογία συσκευών της Texas Instruments		
Οικογένεια επεξεργαστή (Processor Family)	MSP = Ανάμικτων σημάτων επεξεργαστής (Mixed Signal Processor) XMS = Πειραματικού πυριτίου (Experimental Silicon)	
Πλατφόρμα (Platform)	432 = TI's 32-Bit πλατφόρμα μικροελεγκτή χαμηλής κατανάλωσης ισχύος	
Σειρές (Series)	P = Απόδοσης και χαμηλής κατανάλωσης σειρά (Performance and Low-Power Series)	
Σειρά χαρακτηριστικών (Feature Set)	Πρώτο ψηφίο 4 = Flash-based devices μέχρι 48 MHz	Δεύτερο ψηφίο 0 = Γενικού σκοπού
	Τρίτο ψηφίο 1 = ADC14-bits	Τέταρτο ψηφίο R = 256KB of flash 64KB of SRAM M = 128KB of flash 32KB of SRAM
Προαιρετικό: Εύρος θερμοκρασίας (Optional: Temperature Range)	S = 0 °C έως 50 °C I = 40 °C έως 85 °C T = –40 °C έως 105 °C	
Packaging	http://www.ti.com/packaging	
Προαιρετικό: Μορφή διανομής (Optional: Distribution Format)	T = Small reel R = Large reel No markings = Tube or tray	
Προαιρετικά: Πρόσθετα χαρακτηριστικά (Optional: Additional Features)	-EP = Enhanced Product (40°C to 105°C) -HT = Extreme Temperature Parts (55°C to 150°C) -Q1 = Automotive Q100 Qualified	

4ο Κεφάλαιο. Εφαρμογές του MSP432P401R και σύγκριση του με άλλους μικροελεγκτές

Εφαρμογές του MSP432P401R

Ο μικροελεγκτής MSP432P401R λόγω της εξαιρετικά χαμηλής κατανάλωσης ενέργειας και τις υψηλής απόδοσης τον καθιστά ιδανικό για χρήσης φορητών εφαρμογών (Wearable applications) και όχι μόνο, ακολουθούν μερικές εφαρμογές του.

Σε φορητές εφαρμογές (Wearable Applications) να προμηθεύει με PWM τους κατάλληλους μετατροπείς, παράδειγμα σε μετατροπείς ανύψωσης τάσης (DC/DC Boost converters) να μετατρέπουν την χαμηλή τάση ενός φορητού ευκάμπτου οργανικού φωτοβολταικού που βρίσκεται σε ένα μπουφάν σε τάση χρήσιμη για USB (5 V) ώστε να μπορείς να φορτίζεις ένα κινητό.

Εικόνα 17. Μπουφάν με φωτοβολταικά

Εφαρμογές σε δίκτυα αισθητήρων συστημάτων παραγωγής ενέργειας (Energy Harvesting Sensor Network) λόγω τις υπερβολικά υψηλής ακριβείας του ADC που είναι 14bits και 1 MSPS, γίνεται ο έλεγχος διάφορων αισθητήρων όπως θερμοκρασίας, υγρασίας, πυκνότητας ισχύος ηλιακής ακτινοβολίας (πυρανόμετρο), ταχύτητας αέρα (ανεμόμετρο), ατμοσφαιρικής πίεσης κ.α..

Εικόνα 18. Σύστημα παραγωγής ενέργειας, φωτοβολταικά και ανεμογεννήτριες

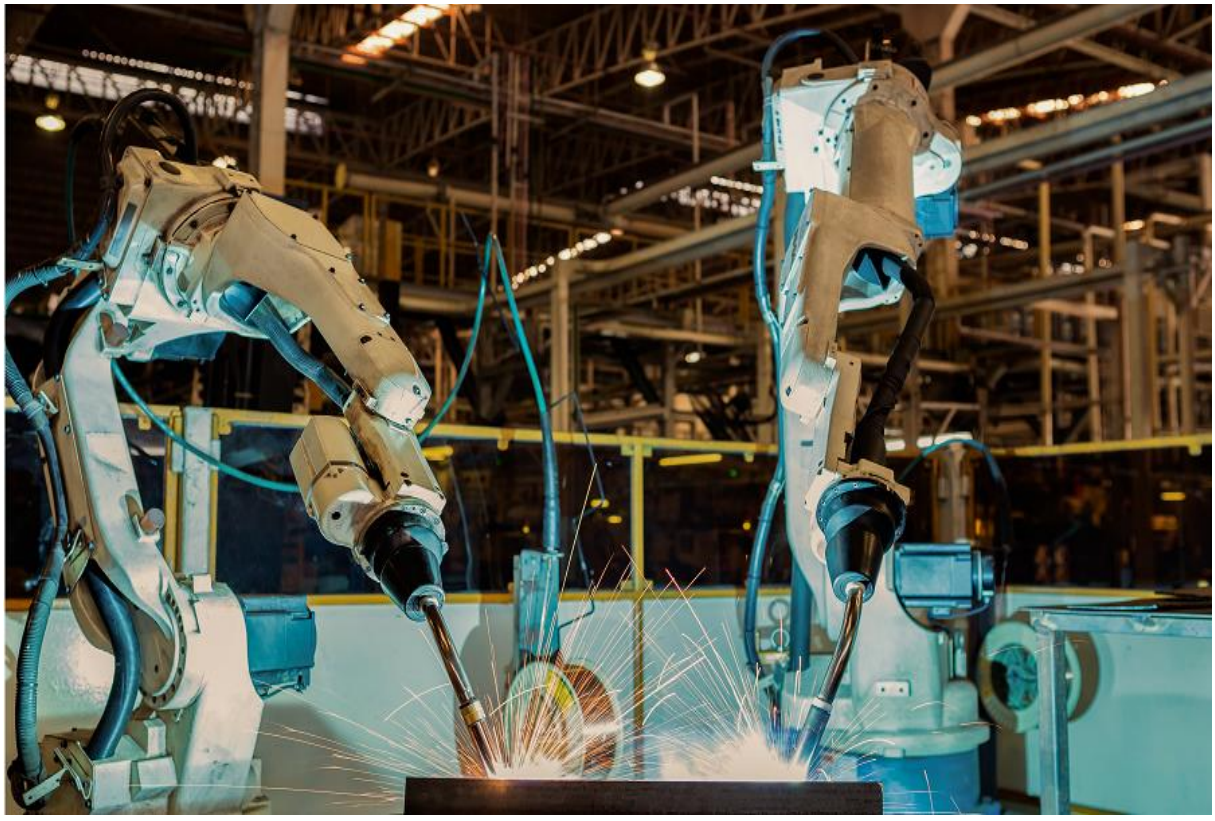
Εικόνα 19. Ανεμόμετρο, Πυρανόμετρο, RTD Pt100

Στην Εικόνα 19. βλέπουμε στην πάνω μεριά αριστερά το Ανεμόμετρο, δεξιά το Πυρανόμετρο και από κάτω τους τον αισθητήρα θερμοκρασίας Pt100.

Σαν ιχνηλάτης εύρεσης μέγιστου σημείου ισχύος [Maximum Power Point Tracker (MPPT)] στα φωτοβολταικά συστήματα.

Συνεπεξεργαστής (Co-Processor) σε έξυπνες συσκευές όπως σαν διανομέας κόμβος αισθητήρων (Sensor hub), ελεγκτής πληκτρολογίου (Keyboard controller), σε εφαρμογές χωρητικού αγγίγματος (Capacitive touch) όπως touchpad, στον έλεγχο φόρτισης μπαταριών και στην διαχείριση ενέργειας τους, στην απτική τεχνολογία (Haptics), σε αισθητήρες προσέγγισης (Proximity Sensors).

Στην βιομηχανία για τον έλεγχο ρομποτικών συστημάτων (Robotics Systems).

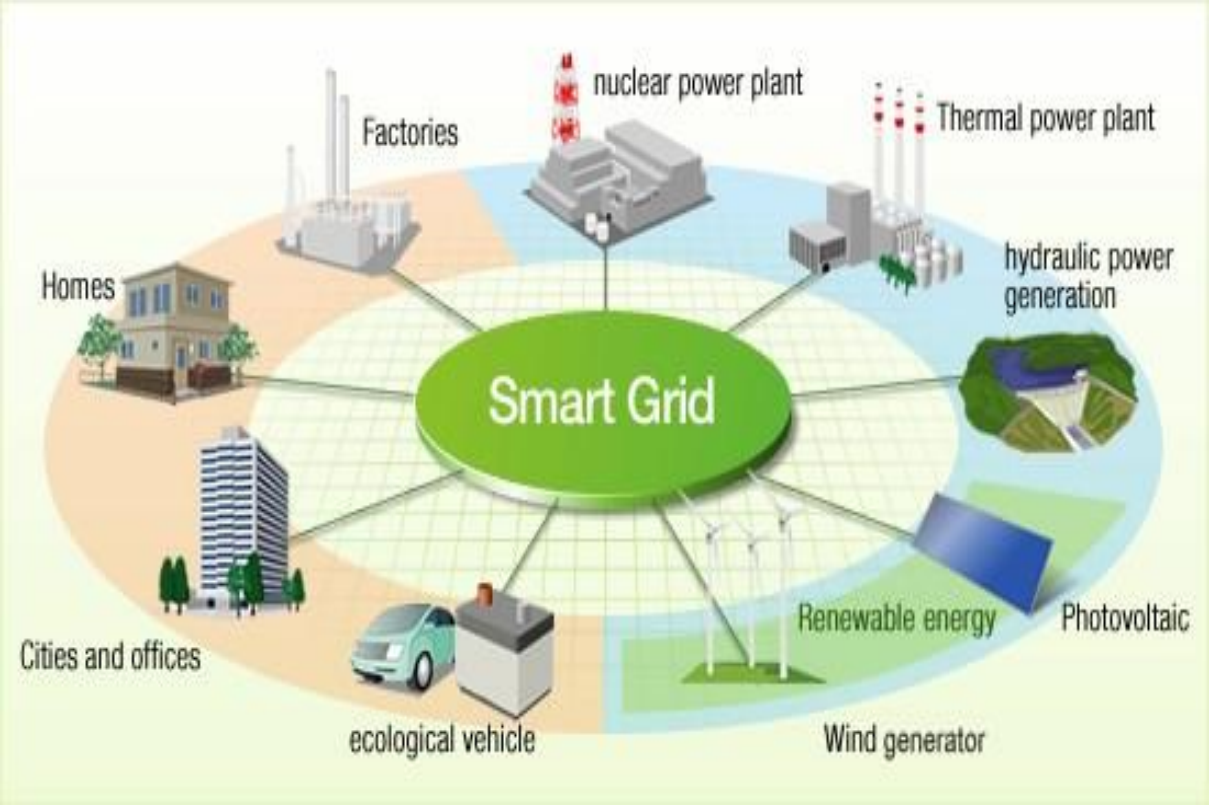


Εικόνα 20. Ρομποτικοί βραχίονες ηλεκτροσυγκόλλησης

Στην βιομηχανία επίσης στα συστήματα συναγερμού και ασφαλείας (Alarm and Security Systems), για την ανθρώπινη διεπαφή (Human Interface) αναγνώριση ήχου ή λέξης, έλεγχο οθόνης αφής (Touch Screen control), καθώς έξυπνη ανίχνευση (Intelligent Sensing) θερμοκρασίας και ρεύματος, ανάλυση υπογραφής (Signature Analysis) για παράδειγμα ανίχνευση θραύσης γυαλιού, επίσης έλεγχος πρόσβασης με ανίχνευση κίνησης, με προσέγγιση και σάρωση δακτυλικού αποτυπώματος (Fingerprint Scan), πυρανίχνευση και ανίχνευση καπνού (Fire and Smoke detection) κ.α.

Στην βιομηχανία για συστήματα ελέγχου κινητήρων ανοιχτού ή κλειστού βρόχου με χρήση PWM.

Στον έλεγχο έξυπνών κτηρίων, δικτύων και πρατήριων υγρών καυσίμων.

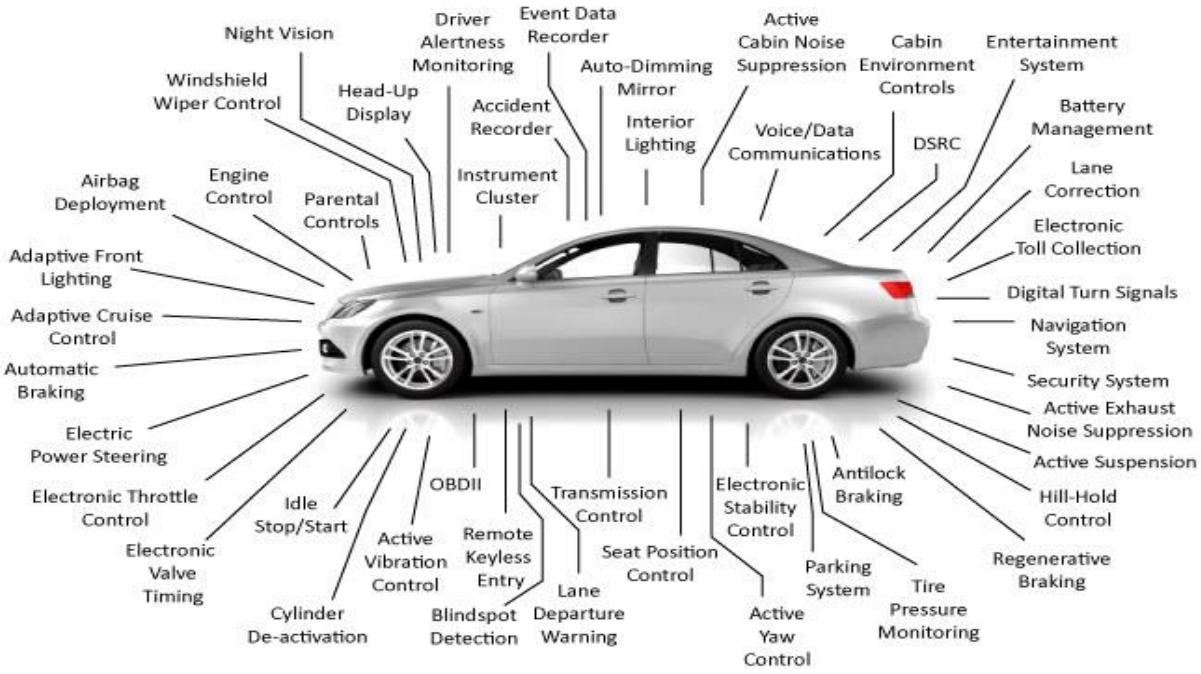


Εικόνα 21. Έξυπνο δίκτυο (Smart grid)

Σε εφαρμογές της ιατρικής (Medical Applications), όπως σε ιατρικά μηχανήματα.

Εικόνα 22. Αιματολογικός αναλυτής

Στον έλεγχο όλων των συστημάτων του αυτοκινήτου και γενικά τον οχημάτων μεταφορών (Automotive Systems).



Εικόνα 23. Συστήματα ελέγχου αυτοκινήτου

Σε απλές οικιακές συσκευές όπως έξυπνα πλυντήρια, ψυγεία, φούρνοι.

Εικόνα 24. Έξυπνο Πλυντήριο, Ψυγείο, Φούρνος

Στην Εικόνα 24. βλέπουμε τις έξυπνές οικιακές συσκευές, στην πάνω μεριά αριστερά το Πλυντήριο, δεξιά το Ψυγείο και από κάτω τους τον Φούρνο.

Στα ηλεκτρονικά ισχύος (Power Electronics) σε DC / DC μετατροπείς ανύψωσης της τάσης (Boost Converter), σε DC / DC μετατροπείς υποβιβασμού τάσης (Buck Converters), απαλούς εκκινητές μηχανών (Soft Starters), DC / AC αντιστροφής (Inverters) κ.α.

Το εύρος εφαρμογών του μικροελεγκτή είναι πολύ μεγάλο, οι μικροελεγκτές γενικά χρησιμοποιούνται σαν "εγκέφαλοι" όλων των ενσωματωμένων συστημάτων (Embedded Systems) και τα ελέγχουν.

Σύγκριση του MSP432P401R με άλλους μικροελεγκτές

Ο μικροελεγκτής MSP432P401R και το αναπτυξιακό MSP-EXP432P401R προτιμήθηκε για την υπερβολικά χαμηλή κατανάλωση ισχύος και την υψηλή απόδοση του καθώς την υπερβολικά υψηλή του ακρίβεια στον ADC οπού είναι 14bits και 1 MSPS.

Τα πλεονεκτήματα του είναι:

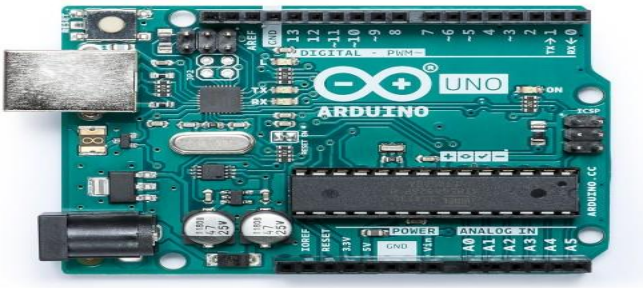
- Χαμηλής κατανάλωση στα 80 μA/MHz (<100μA / MHz) και έχει υψηλής απόδοσης επεξεργαστή ARM 32-Bit Cortex-M4F με FPU και MPU
- Υψηλή συχνότητα ρολογιού (Clock Speed Frequency) έως 48 MHz
- 256 KB flash memory, 64 KB SRAM, 32 KB ROM
- Λειτουργίες εξαιρετικά χαμηλής κατανάλωσης (π.χ. LPM4.5 στα 25 nA), καθώς λειτουργίες ύπνου και βαθύ ύπνου
- ADC 14bits 1 MSPS
- 2x32bits Timers
- 4x16bits Timers ο καθένας με δυνατότητες επί 5 Capture, Compare, PWM
- 84 είσοδοι και έξοδοι γενικού σκοπού [General Purpose Input Output(GPIO)], 10 θύρες (Ports) από 8bits (8 pins) η κάθε μία, όλες οι θύρες είναι ψηφιακές
- 24 Αναλογικά κανάλια
- Προγραμματιζόμενες Pull-Up και Pull-Down αντιστάσεις σε όλες τις θύρες
- Όλες οι I/Os έχουν δυνατότητα χωρητικού αγγίγματος (Capacitive Touch)
- 48 I/Os με δυνατότητα Interrupt λειτουργίας και ξυπνήματος (Wake Up)
- 24 I/O με δυνατότητα χαρτογράφησης (Port Mapping Capability)
- 8 Serial I/O, 8 SPI, 4 UART, 4 I^2C
- 8 Direct Memory Access (DMA)
- Δυνατότητα Multitasking (Λόγω του 3-Stage Pipeline)
- Φτηνό κόστος αγοράς για αυτά που προσφέρει (Value for money),12.99$ ή 11.22€ όλο το αναπτυξιακό MSP-EXP432P401R από την ίδια την Texas Instruments

Τα μειονεκτήματα του είναι:

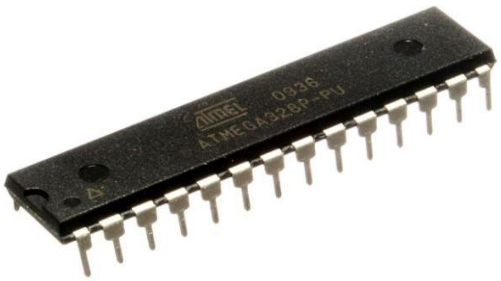
- Αρκετά περίπλοκος μικροελεγκτής (Complexity) στο να προγραμματιστεί δεν είναι φιλικός προς τους αρχάριους χρήστες και για τους έμπειρους για να αλλάξουν κάτι απλό θέλει αρκετή διαδικασία σε σύγκριση για το ίδιο πράγμα σε έναν Arduino
- Αδύναμες ψηφιακές θύρες σε κανονική ρύθμιση δύναμης του οδηγού (Regular Drive Strength) το ρεύμα εξόδου είναι 6 mA, καθώς μόνο 4 I/O (οι P2.0, P2.1, P2.2, P2.3) σε αυξημένη ρύθμιση δύναμης (High Drive Strength) έχουν 20 mA ρεύμα εξόδου, καθώς πάντα η μέγιστη τάση εξόδου σε όλες τις περιπτώσεις είναι ~3.3 V

Θα συγκρίνουμε το αναπτυξιακό MSP-EXP432P401R με το διάσημο αναπτυξιακό της Microchip (Atmel) AVR τον Arduino UNO Rev3 βασισμένο στον μικροελεγκτή ATmega328P (8-bits RISC Processor).

Ο λόγος που συγκρίνουμε με το MSP-EXP432P401R με τον Arduino UNO Rev3 είναι γιατί το Arduino είναι μια πλατφόρμα που χρησιμοποιείται από τους ποιό αρχάριους χρήστες έως τους ποιό εξειδικευμένους επαγγελματίες για τον λόγο ευκολίας προγραμματισμού του και για τον λόγο ότι το Ολοκληρωμένο Περιβάλλον Ανάπτυξης [Integrated Development Environment (IDE)] του είναι ελεύθερα προσβάσιμο (Open Source) και αρκετά κατανοητό στην χρήση του που το καθιστά ιδανικό για όλους τους χρήστες.



Εικόνα 25. Arduino UNO Rev3



Εικόνα 26. Τσιπ μικροελεγκτή ATmega328P

Πίνακας 2. Συγκριτικός πίνακας MSP-EXP432P401R με Arduino UNO Rev3		
Αναπτυξιακά	MSP-EXP432P401R	Arduino UNO Rev3
Ονομασία τσιπ μικροελεγκτή	MSP432P401R	ATmega328P
Τάση λειτουργίας MCU	3.3 V	5 V
Κατανάλωση ρεύματος σε Active mode	80 μA/MHz (στα 3.3 V), μέγιστη κατανάλωση 3.84 mA	~2.9 mA/MHz (στα 5 V), μέγιστη κατανάλωση 46.5 mA
MCU Bits	32	8
Ανάλυση ADC σε Bits	14	10
Clock Speed	48 MHz	16 MHz
Flash Memory	256 KB	32 KB
SRAM	64 KB	2 KB
ROM	32 KB	1 KB
Digital I/O Pins (Από τις θύρες συνδέσεων)	33	14
Analog Input Pins (Από τις θύρες συνδέσεων)	8	6
Ρεύμα ανά I/O Pins	6 mA (Regular Drive) / 20 mA (High Drive)	20 mA (στα 5 V) / 50 mA (στα 3.3 V)
Κόστος αγοράς σε ευρώ	11.22 €	22 €

5ο Κεφάλαιο. Οδηγός χρήσης του Code Composer Studio (CCS) IDE και ανάλυση βασικών καταχωρητών (Registers)

Οδηγός χρήσης του CCSv8 IDE

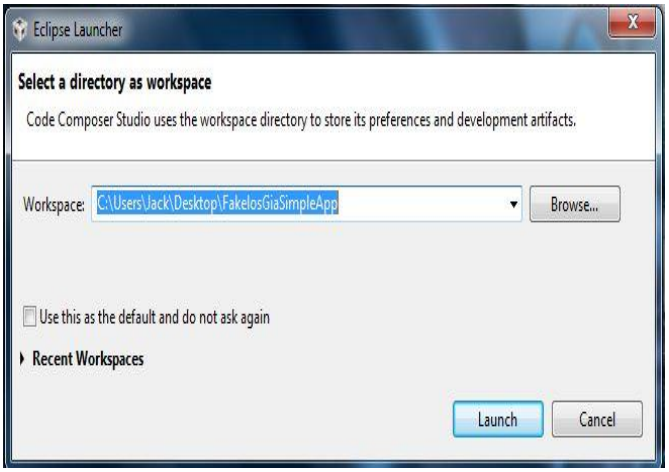
Σε αυτό το κεφάλαιο θα αναλυθούν τα βασικά εργαλεία του Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης (IDE) της Texas Instruments το Code Composer Studio IDE, είναι ένα επαγγελματικό εργαλείο με πάρα πολλές δυνατότητες. Εμπεριέχεται και η λειτουργία του Energy Trace καθώς το πρόγραμμα έχει και λειτουργία απασφαλμάτωσης (Debug Mode).

Εικόνα 27. Code Composer Studio IDE v8

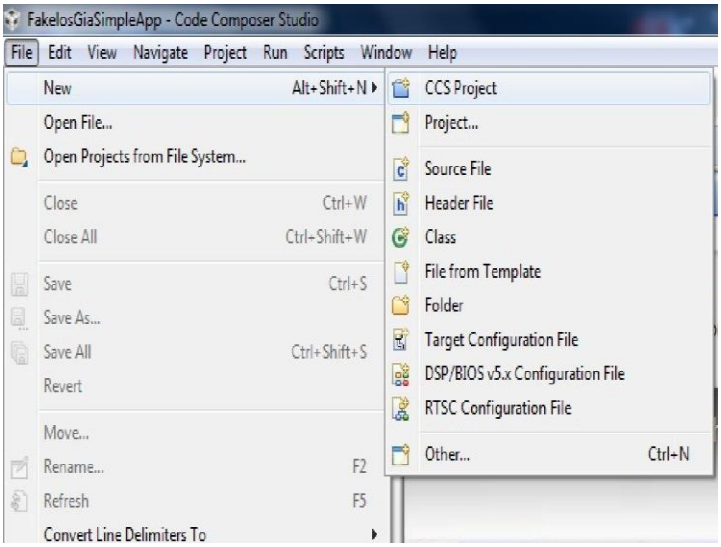
Δημιουργία απλής εφαρμογής και φόρτωσης της στο αναπτυξιακό MSP-EXP432P401R σε γλώσσα προγραμματισμού C. Η διαδικασία δημιουργίας της εφαρμογής θα αναλυθεί βήμα προς βήμα, καθώς θα γίνει και μια περιήγηση των λειτουργιών του CCSv8 IDE.

1. Πρώτα συνδέεται μέσω του USB, τον μικροελεγκτή στον υπολογιστή
2. Ανοίγετε το CCS 

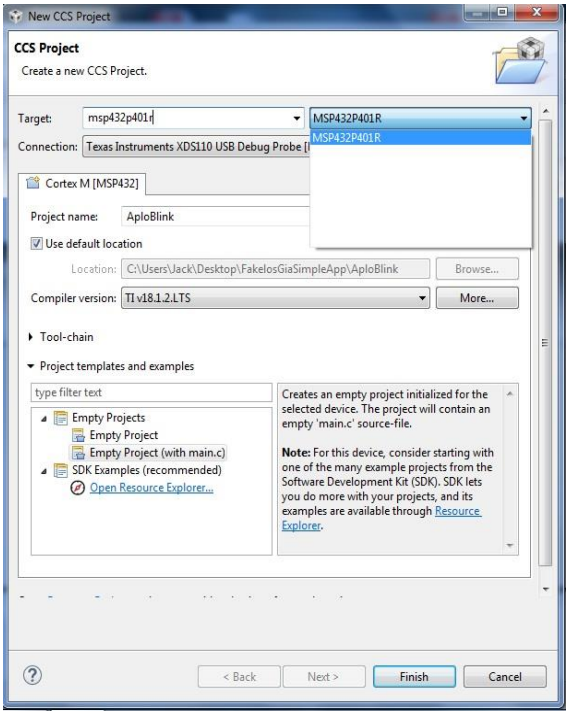
3. Ζητείτε διεύθυνση Workspace από το CCS, να δημιουργηθεί ή να του δοθεί ένας φάκελος που θα αποθηκεύσει της ρυθμίσεις (Preferences) και τα Projects που θα δημιουργηθούν, δίνεται η διεύθυνση και το όνομα του φακέλου και πατάτε το Launch



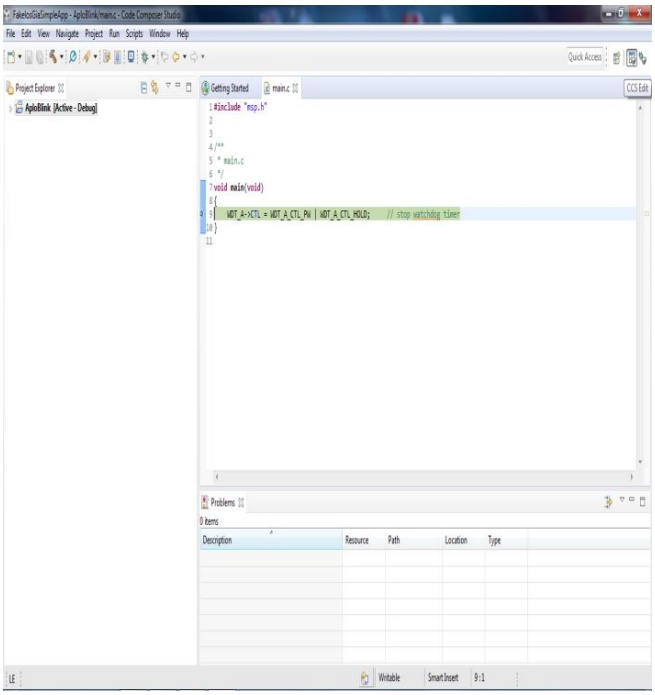
4. Έπειτα δημιουργείτε ένα νέο CCS Project, File > New > CCS Project



5. Στην συνέχεια ανοίγει το παράθυρο δημιουργίας του New CCS Project και
   - δίνεται όνομα στο Project στην επιλογή Project name:
   - καθώς γίνεται η επιλογή οικογενείας της συσκευής (Device Family) γράφοντας στο Target: και επιλέγοντας από τα δεξιά την κατάλληλη συσκευή
   - αποθηκεύεται στον φάκελο που δημιουργήθηκε το Workspace (Use default location)
   - επιλέγεται από το Project templates and examples το Empty Project (with main.c) και πατάτε Finish

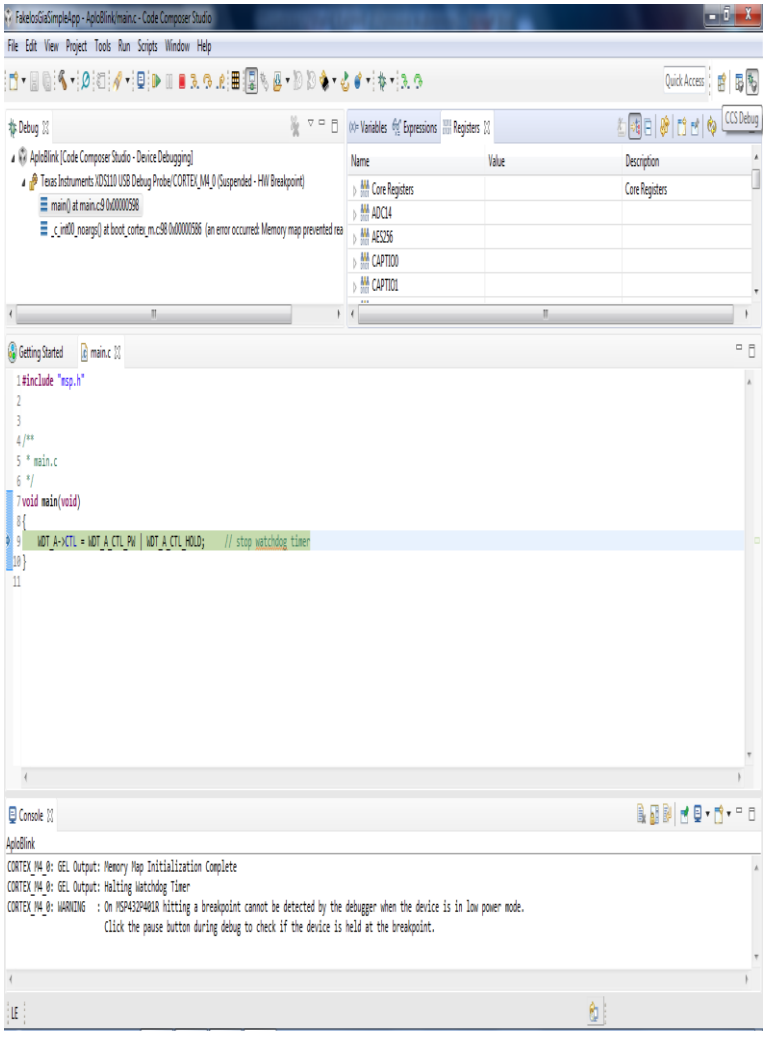


6. Μετά το Finish δημιουργεί το πρόγραμμα και εμφανίζετε σε λειτουργία CCS edit όπως φαίνεται στην Εικόνα 28. δεξιά πάνω , στην μέση δεξιά βρίσκεται η main.c εδώ γράφεται το κυρίως πρόγραμμα (Εφαρμογή), επίσης έχει συμπεριλάβει (Include) την κυρίως βιβλιοθήκη "msp.h" που εμπεριέχει όλους τους καταχωρητές (registers) και τις εντολές που χρειάζεται για να προγραμματιστεί ο μικροελεγκτής, στα αριστερά βρίσκεται ο περιηγητής του Project με την ονομασία Project Explorer που μας δείχνει ποιό Project βρίσκεται σε λειτουργία και σε τη λειτουργία (Active ή Debug), στο κάτω μέρος της Εικόνας 28. φαίνονται τα προβλήματα που μπορεί να δημιουργηθούν (Problems) στο πρόγραμμα και η αναλυτική περιγραφή τους, στην συγκεκριμένη μπάρα με τα προβλήματα εμφανίζονται και άλλες λειτουργιές του CCS όταν ενεργοποιηθούν όπως το Energy Trace, Serial Monitor κ.α.

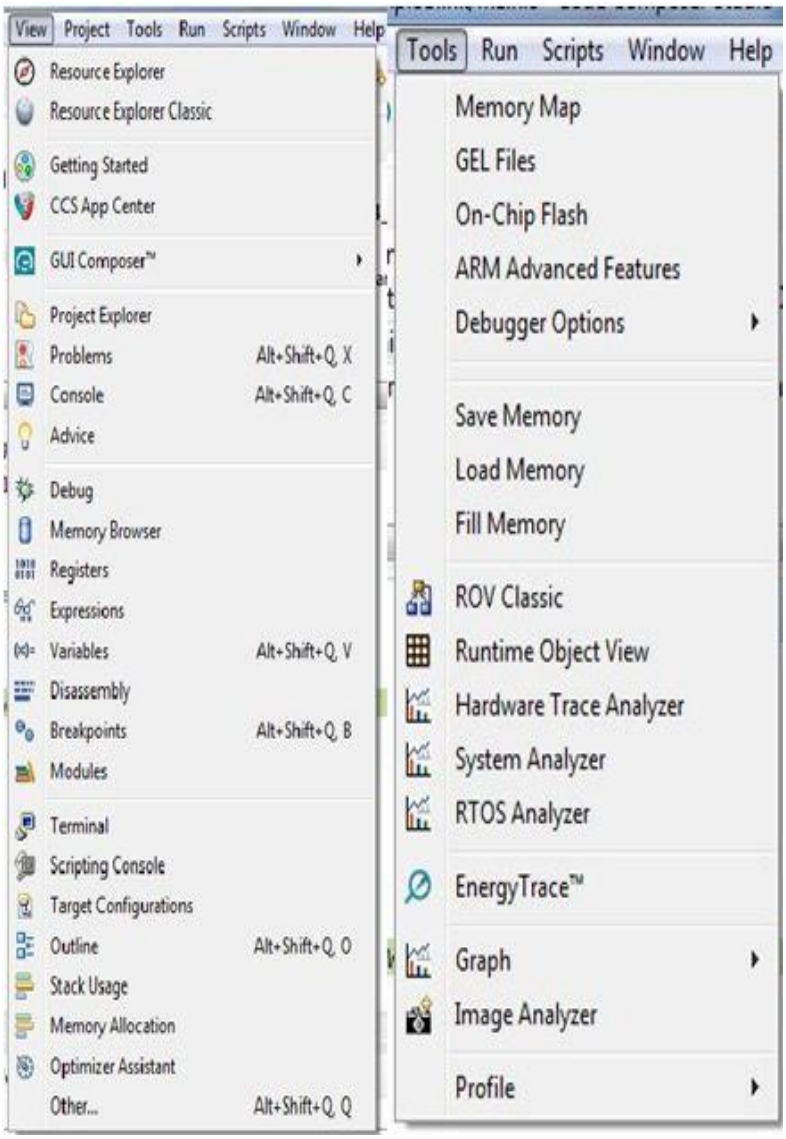


Εικόνα 28. CCS Edit Mode

7. Πατώντας το κουμπί του Debug  μπαίνει το CCS σε λειτουργία απασφαλμάτωσης (CCS Debug Mode) όπως φαίνεται στην Εικόνα 29. σε αυτή την λειτουργία δίνονται πάρα πολλά εργαλεία έλεγχου και απασφαλμάτωσης, όπως έλεγχος μεταβλητών, εκφράσεων, καταχωρητών (Variables, Expressions, Registers), εμφανίζονται διάφορα εργαλεία (Tools) μόνο σε Debug Mode (η επιλογή των Modes γίνεται από τα κουμπιά δεξιά πάνω π.χ. επιστροφή σε CCS edit Mode), στην Εικόνα 30. βλέπουμε την πληθώρα εργαλείων του CCS



Εικόνα 29. CCS Debug Mode



Εικόνα 30. CCS View και Tools

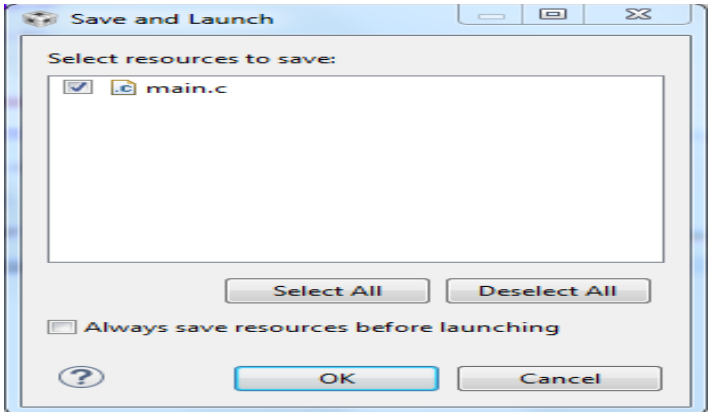
8. Στην συνέχεια βγαίνουμε από το debug mode πατώντας το κουμπί terminate  και γράφουμε το πρόγραμμα μας που θα αναβοσβήνει (Blink) ένα Led, οι καταχωρητές και τα μασκαρίσματα που χρησιμοποιούνται στον παρακάτω κώδικα που ακολουθεί θα αναλυθούν στην ενότητα ανάλυσης των βασικών καταχωρητών του MSP432P401R, θα γίνει η ανάλυση και στα βασικά μασκαρίσματα (Masking) AND, OR, XOR γιατί είναι ο βασικός τρόπος για να κάνουμε αλλαγές στους καταχωρητές

Κώδικας CCS σε γλώσσα προγραμματισμού C:

```
#include "msp.h"

unsigned int i;
/**
 * main.c
 */
void main(void)
{
	WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD; // stop watchdog timer
	P2DIR |= 0b10000000; //P2.7 Eksodos

	while(1)
	{
	    P2OUT ^= 0b10000000; //Enallagi (Toggle) Eksodou P2.7

	    for(i=0; i<20000; i++) //ena mikro delay
	    {

	    }
	}
}
```

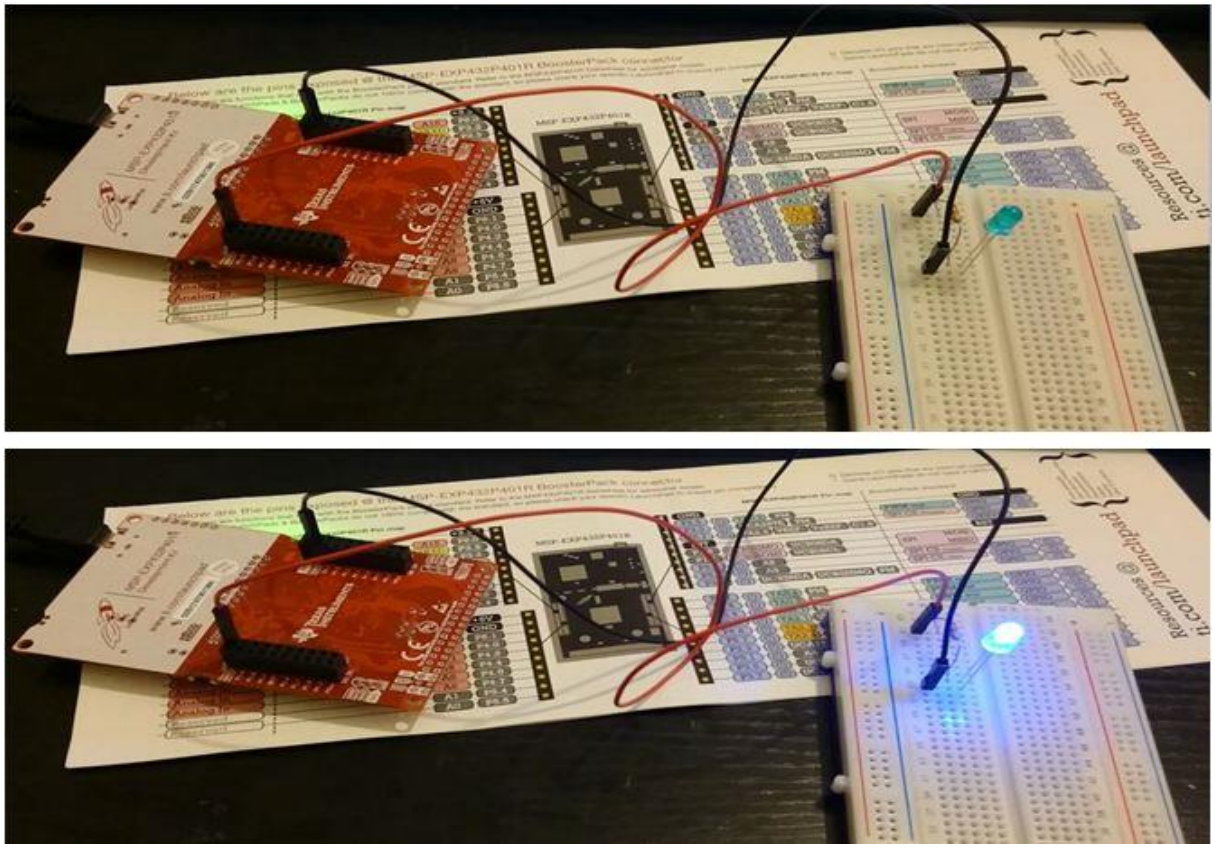
9. Έπειτα φορτώνουμε (Upload) το πρόγραμμα στην Flash memory του μικροελεγκτή μας πατώντας το κουμπί Flash 

, αρχίζει πρώτα να το κάνει Compile (μετατροπή του κώδικα που γράψαμε, από γλώσσα C σε γλώσσα μηχανής) και έπειτα να το φορτώνει στην Flash μνήμη του μικροελεγκτή, όσο φορτώνεται το πρόγραμμα στην μνήμη του μικροελεγκτή ενδεικτικά αναβοσβήνει το κόκκινο LED πάνω στο αναπτυξιακό MSP-EXP432P401R, υπάρχει και το σκέτο build (compile) χωρίς το φόρτωμα στον μικροελεγκτή πατώντας το κουμπί του build 
10. Εάν γίνει αλλαγή στον κώδικά και θέλουμε να το ξαναφορτώσουμε, πατάμε πάλι το κουμπί Flash αυτή την φόρα μας ανοίγει ένα παράθυρο με το όνομα Save and Launch και διαλέγουμε την main.c που εμφανίζεται και πατάμε OK



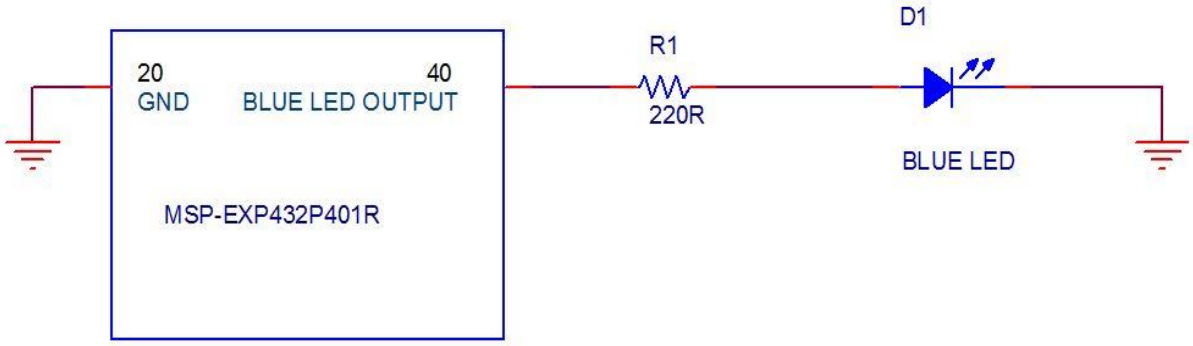
11. Όταν σταματήσει να αναβοσβήνει το κόκκινο LED στο αναπτυξιακό τότε το πρόγραμμα έχει φορτωθεί επιτυχώς στον μικροελεγκτή

Το πρόγραμμα που φτιάξαμε με την ονομασία AploBlink σε λειτουργία αναβοσβήνοντας ένα μπλε LED.



Εικόνα 31. Λειτουργία προγράμματος AploBlink

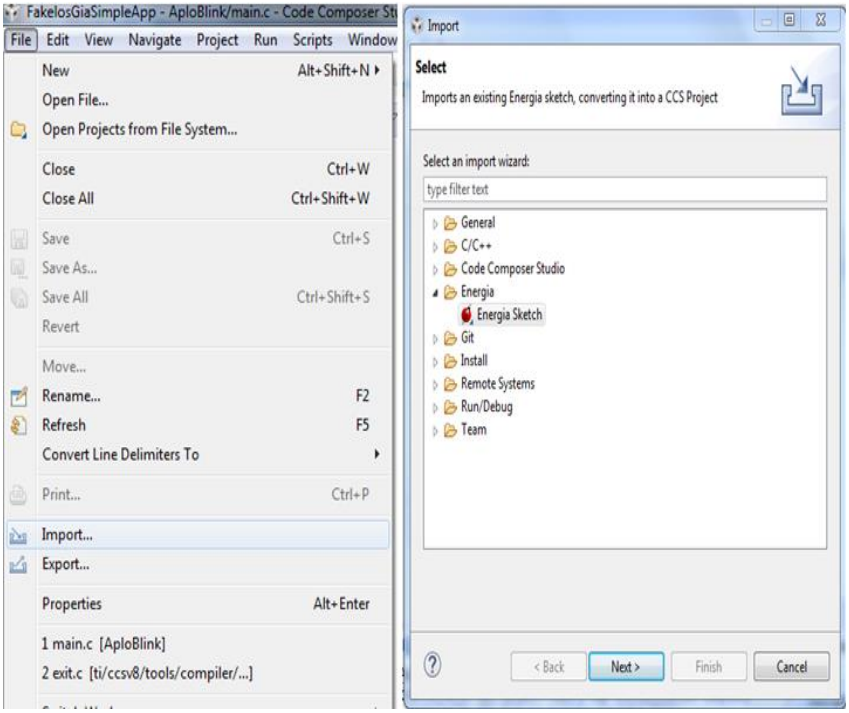
Το σχηματικό με τα Pinout (σε νούμερα από το 1 έως το 40 δηλαδή από τις θύρες συνδέσεων J1 έως J4).

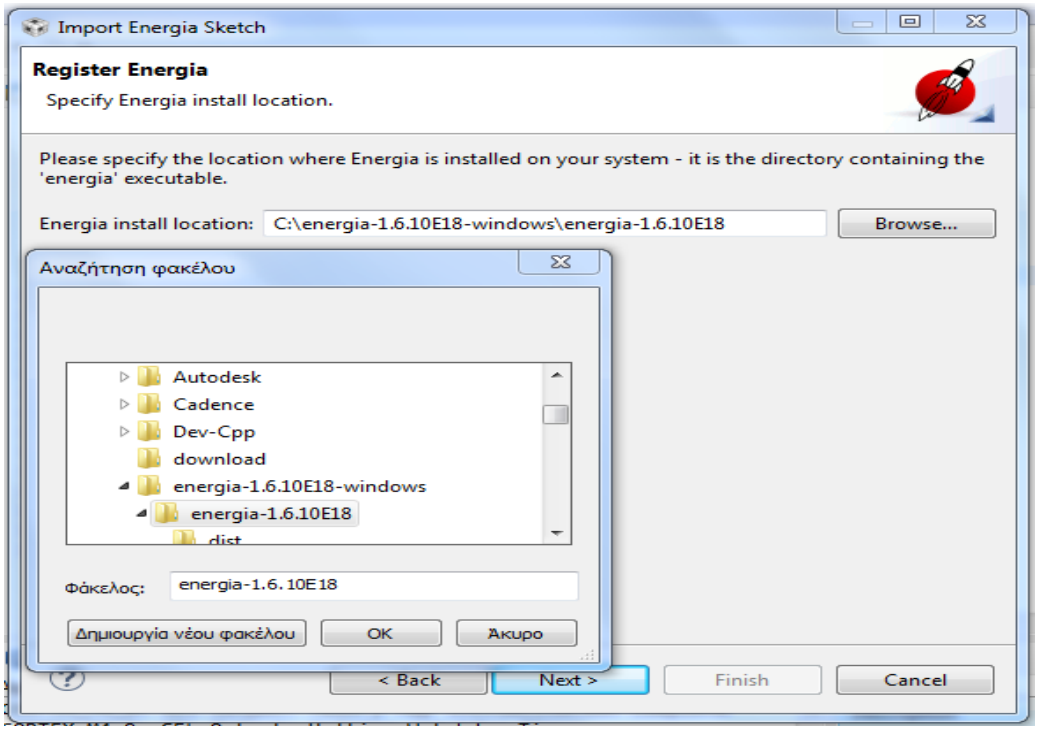


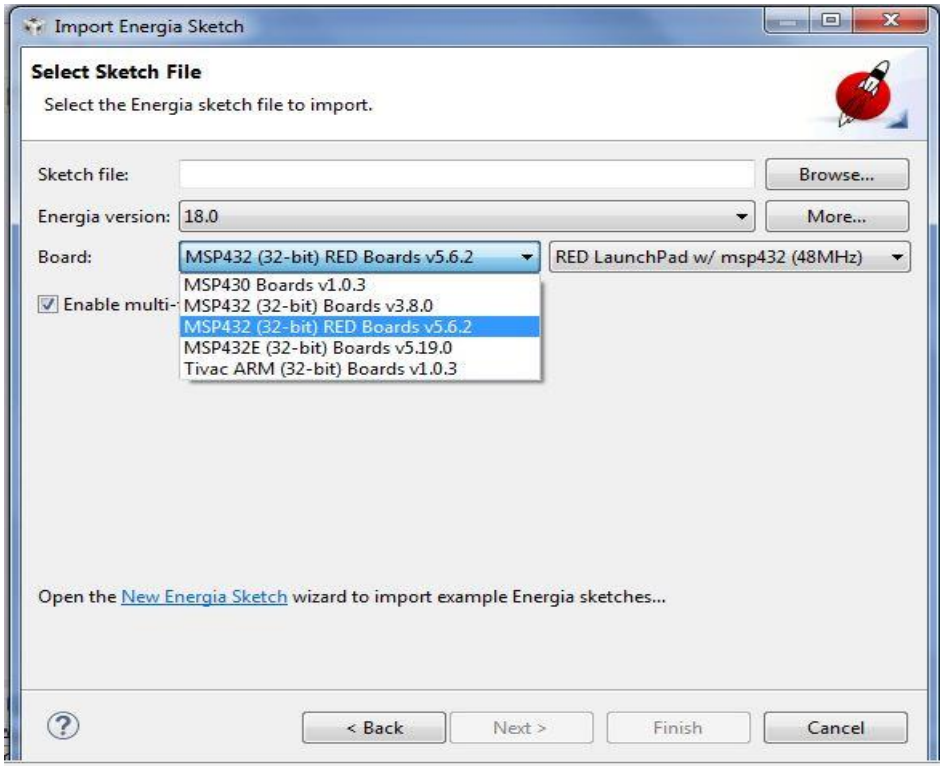
Σχηματικό 1. AploBlink

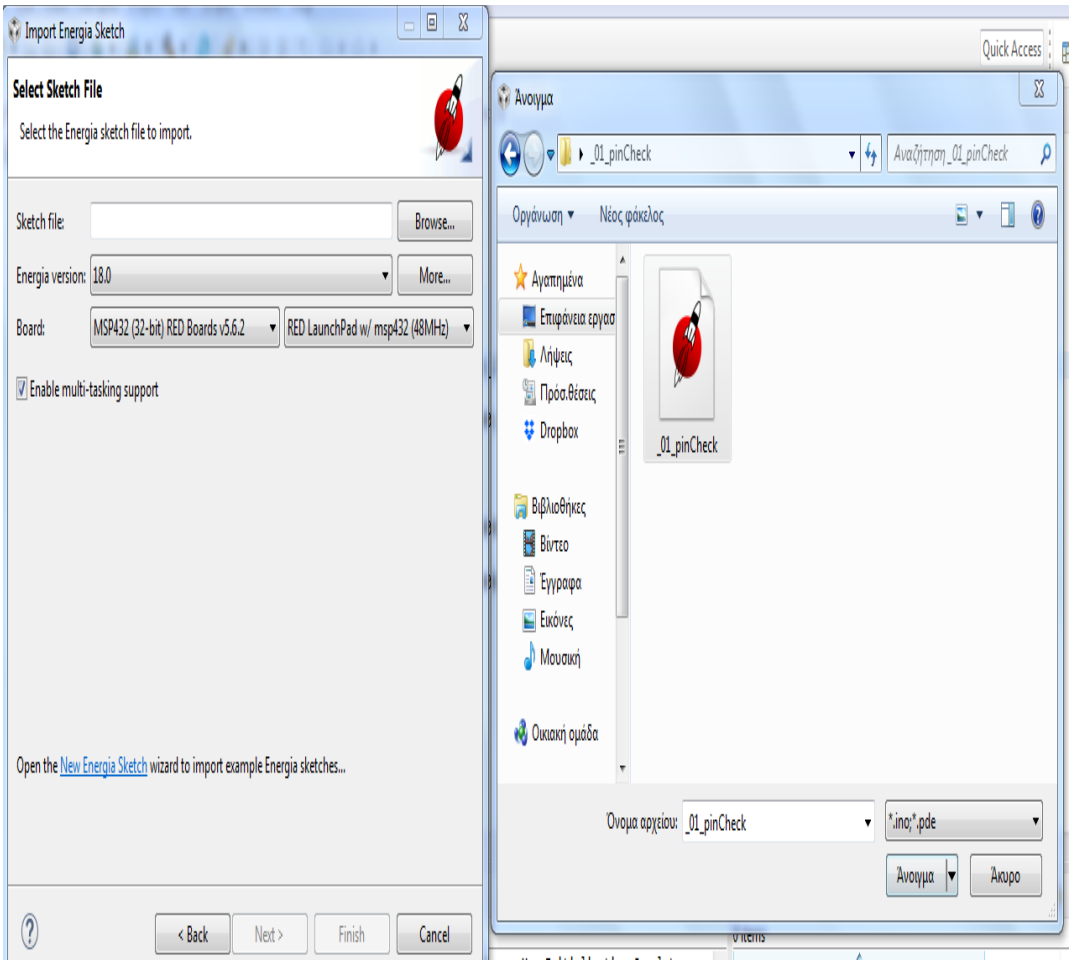
Εισαγωγή προγράμματος από το Energia IDE

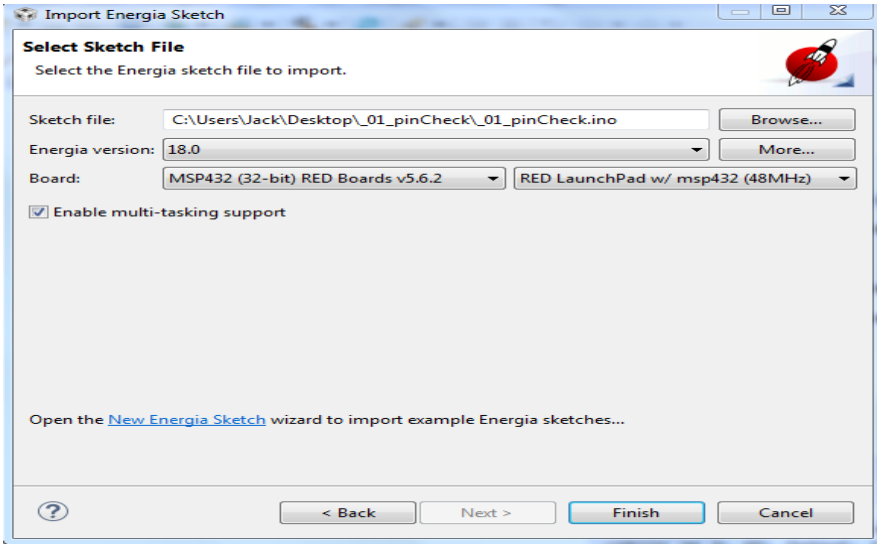
Στην συνέχεια βλέπουμε την εισαγωγή (Import) προγράμματος του Energia IDE στο CCS IDE (δίνει αυτή την δυνατότητα) File > Import > Energia Sketch > Search το Install του Energia IDE > επιλογή αναπτυξιακού (board) > επιλογή Energia Sketch File > Finish.



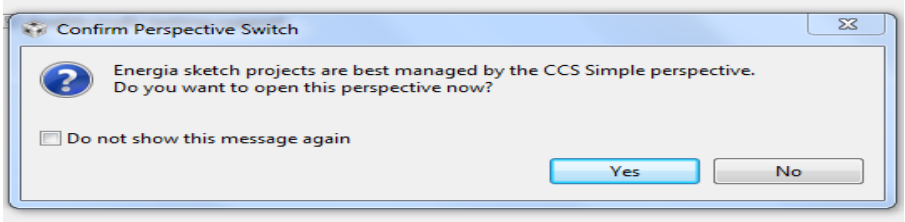



Import Energia Sketch
Select Sketch File
Select the Energia sketch file to import.
Sketch file:
Browse...
Energia version:
18.0
More...
Board:
MSP432 (32-bit) RED Boards v5.6.2
RED LaunchPad w/ msp432 (48MHz)
MSP430 Boards v1.0.3
MSP432 (32-bit) Boards v3.8.0
MSP432 (32-bit) RED Boards v5.6.2
MSP432E (32-bit) Boards v5.19.0
Tivac ARM (32-bit) Boards v1.0.3
Enable multi-
Open the New Energia Sketch wizard to import example Energia sketches...
< Back
Next >
Finish
Cancel

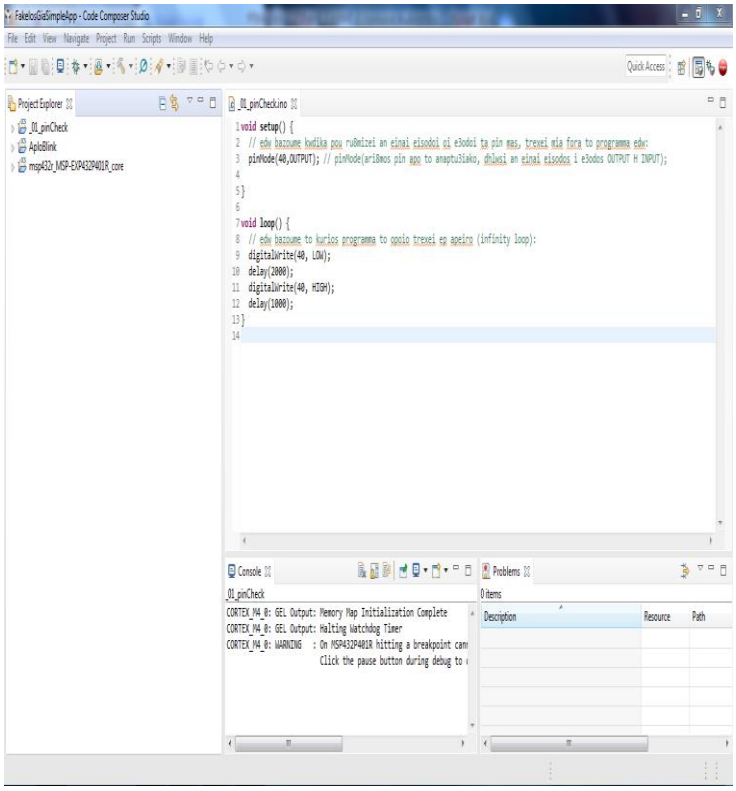

Import Energia Sketch
Select Sketch File
Select the Energia sketch file to import.
Sketch file:
Browse...
Energia version:
18.0
More...
Board:
MSP432 (32-bit) RED Boards v5.6.2
RED LaunchPad w/ msp432 (48MHz)
Enable multi-tasking support
Open the New Energia Sketch wizard to import example Energia sketches...
< Back
Next >
Finish
Cancel
Quick Access
Άνοιγμα
_01_pinCheck
Αναζήτηση _01_pinCheck
Οργάνωση
Νέος φάκελος
Αγαπημένα
Επιφάνεια εργασ
Λήψεις
Πρόσ.θέσεις
Dropbox
Βιβλιοθήκες
Βίντεο
Έγγραφα
Εικόνες
Μουσική
Οικιακή ομάδα
_01_pinCheck
Όνομα αρχείου: _01_pinCheck
.ino;.pde
Άνοιγμα
Άκυρο



Πατάμε Yes



και τώρα έχει γίνει εισαγωγή (import) το sketch του Energia IDE στο CCS IDE όπως φαίνεται και στην Εικόνα 32.



Εικόνα 32. Energia Sketch

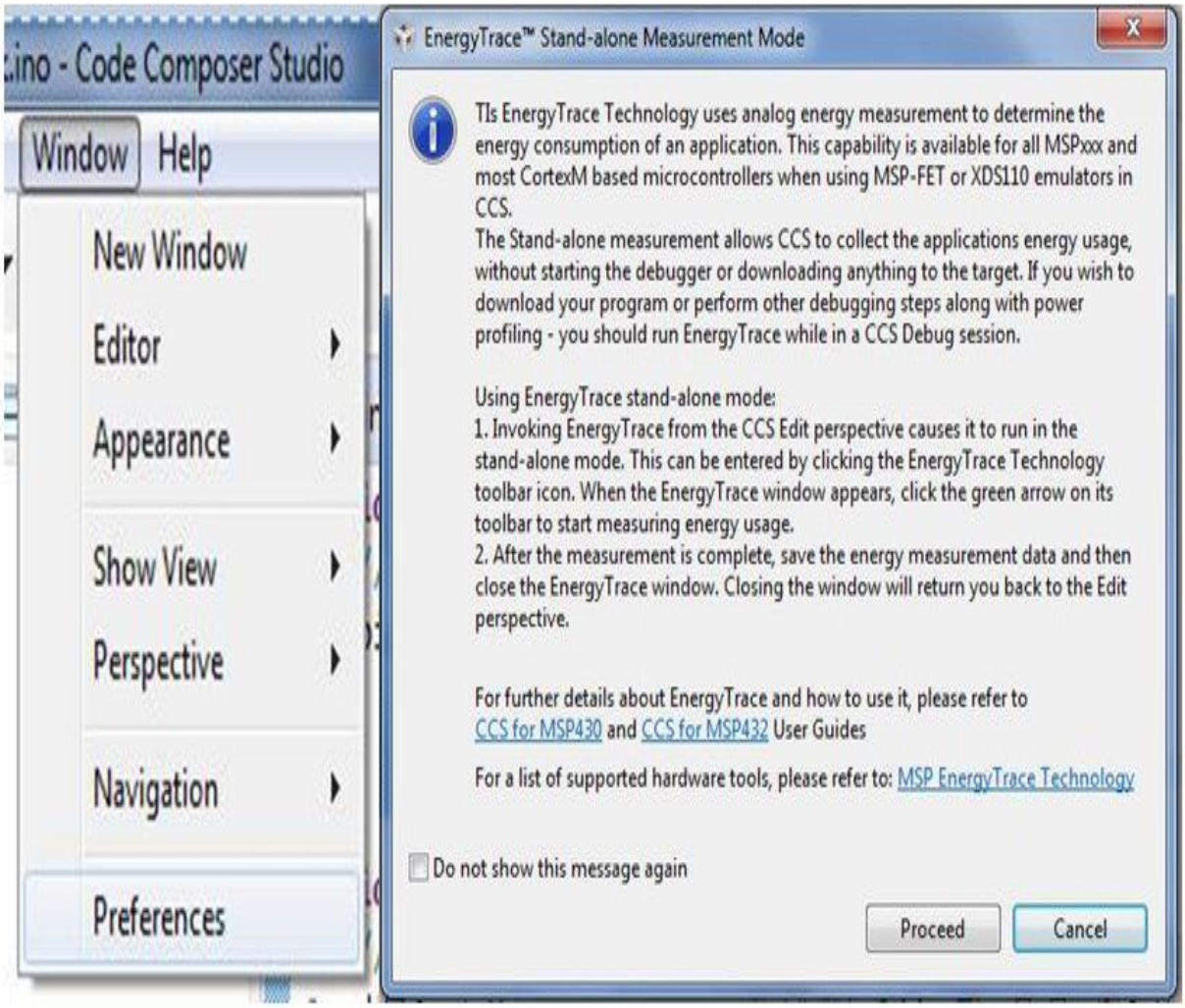
Λειτουργία του Energy Trace

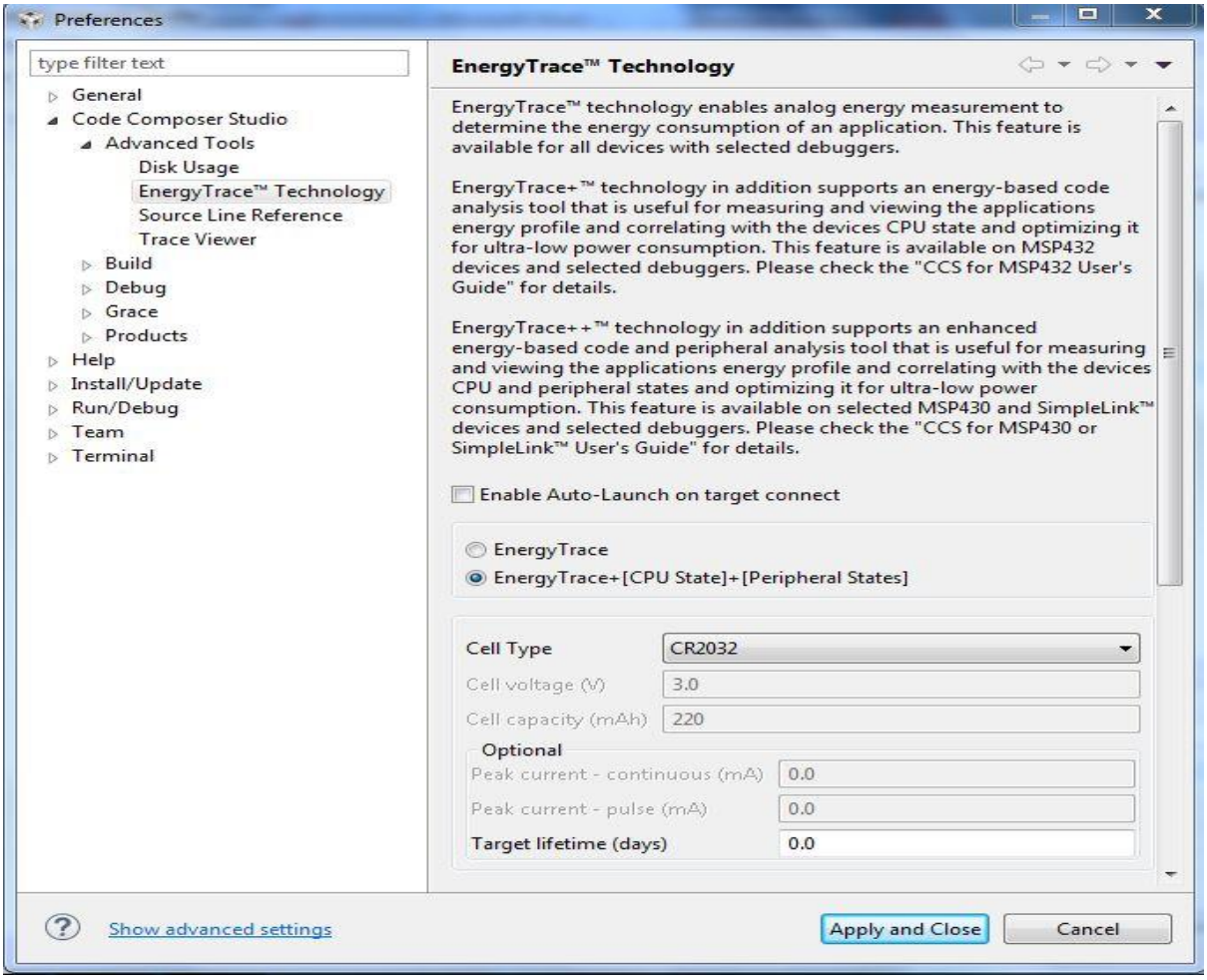
Για να ενεργοποιηθεί το Energy Trace πρώτη φορά ακολουθείτε η διαδικασία ρυθμίσεων του: Window > Preferences > από τα αριστερά του παράθυρου Preferences επιλέγεται Code Composer Studio > Advanced Tools > Energy Trace Technology , στην συνέχεια δεξιά του παραθύρου Preferences γίνονται οι επιλογές:

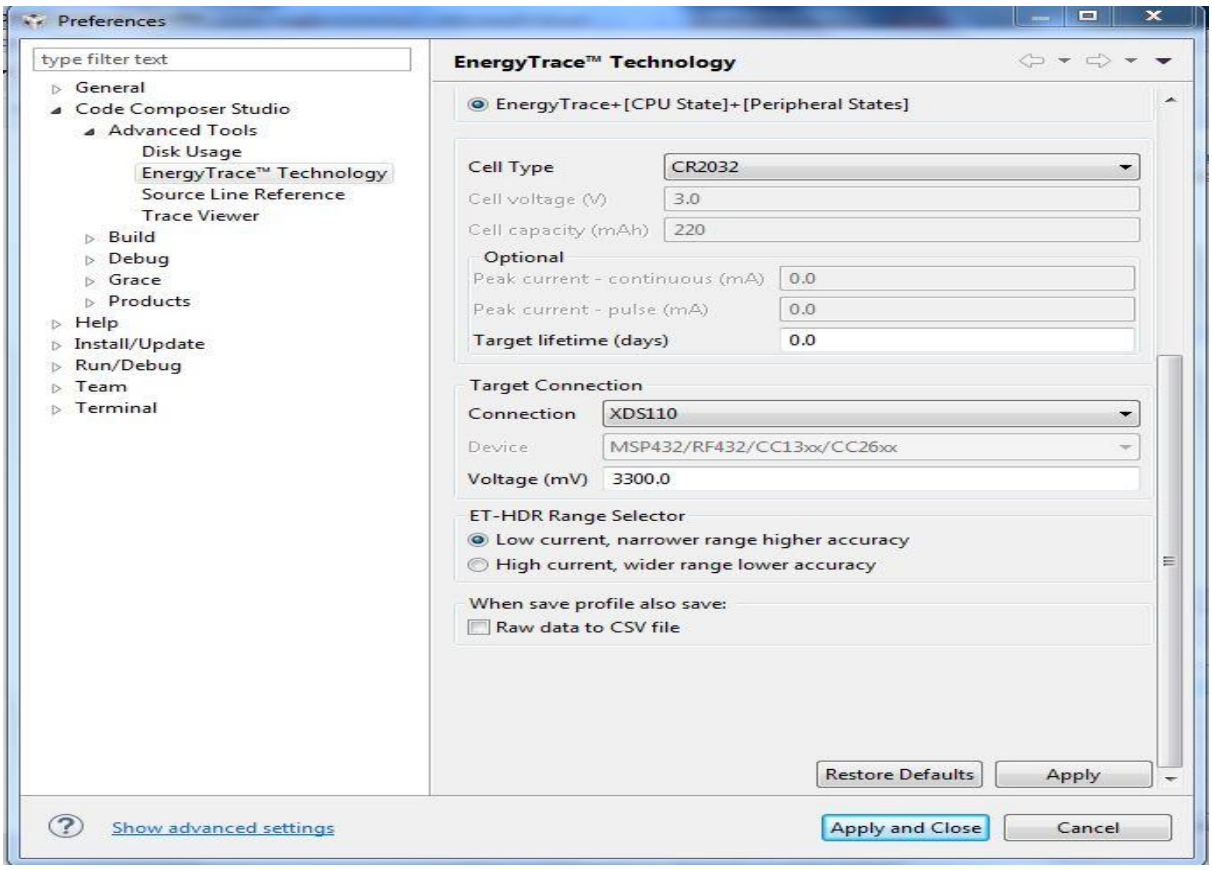
- Energy/Trace+[CPU State]+[Peripheral States]
- Connection: XDS110

Πατάμε το Apply and Close και μπαίνει αυτόματα στο CCS Debug Mode στο οποίο τέρμα κάτω δεξιά εμφανίζετε το Energy Trace.

Για την ενεργοποιήσει του Energy Trace τις επόμενες φόρες πατάμε απλώς το κουμπί με το εικονίδιο του Energy Trace  και πατάμε Proceed στο παράθυρο που μας ανοίγει έπειτα (EnergyTrace Stand-alone Measurement Mode) και μας ξαναβάζει στο CCS Debug Mode όπως την πρώτη ενεργοποίηση του.



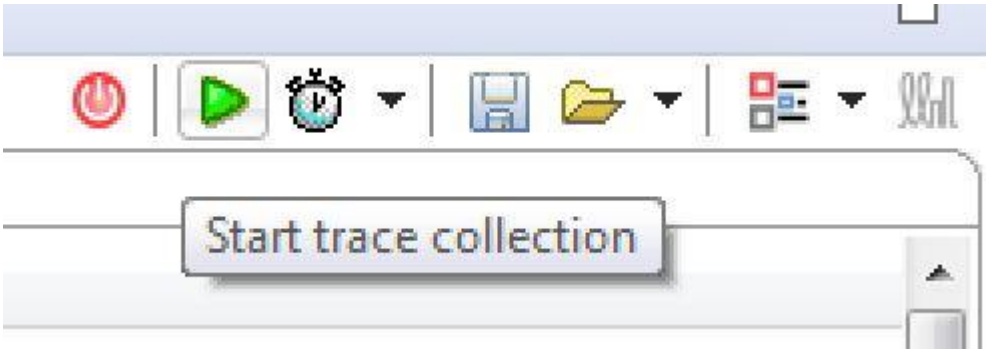

Preferences
type filter text
General
Code Composer Studio
Advanced Tools
Disk Usage
EnergyTrace™ Technology
Source Line Reference
Trace Viewer
Build
Debug
Grace
Products
Help
Install/Update
Run/Debug
Team
Terminal
EnergyTrace™ Technology
EnergyTrace™ technology enables analog energy measurement to determine the energy consumption of an application. This feature is available for all devices with selected debuggers.
EnergyTrace+™ technology in addition supports an energy-based code analysis tool that is useful for measuring and viewing the applications energy profile and correlating with the devices CPU state and optimizing it for ultra-low power consumption. This feature is available on MSP432 devices and selected debuggers. Please check the "CCS for MSP432 User's Guide" for details.
EnergyTrace++™ technology in addition supports an enhanced energy-based code and peripheral analysis tool that is useful for measuring and viewing the applications energy profile and correlating with the devices CPU and peripheral states and optimizing it for ultra-low power consumption. This feature is available on selected MSP430 and SimpleLink™ devices and selected debuggers. Please check the "CCS for MSP430 or SimpleLink™ User's Guide" for details.
Enable Auto-Launch on target connect
EnergyTrace
EnergyTrace+[CPU State]+[Peripheral States]
Cell Type
CR2032
Cell voltage (V)
3.0
Cell capacity (mAh)
220
Optional
Peak current - continuous (mA)
0.0
Peak current - pulse (mA)
0.0
Target lifetime (days)
0.0
Show advanced settings
Apply and Close
Cancel

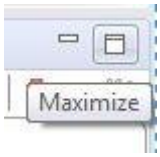

Preferences
type filter text
General
Code Composer Studio
Advanced Tools
Disk Usage
EnergyTrace™ Technology
Source Line Reference
Trace Viewer
Build
Debug
Grace
Products
Help
Install/Update
Run/Debug
Team
Terminal
EnergyTrace™ Technology
EnergyTrace+[CPU State]+[Peripheral States]
Cell Type
CR2032
Cell voltage (V)
3.0
Cell capacity (mAh)
220
Optional
Peak current - continuous (mA)
0.0
Peak current - pulse (mA)
0.0
Target lifetime (days)
0.0
Target Connection
Connection
XDS110
Device
MSP432/RF432/CC13xx/CC26xx
Voltage (mV)
3300.0
ET-HDR Range Selector
Low current, narrower range higher accuracy
High current, wider range lower accuracy
When save profile also save:
Raw data to CSV file
Restore Defaults
Apply
Show advanced settings
Apply and Close
Cancel

Στο Energy Trace

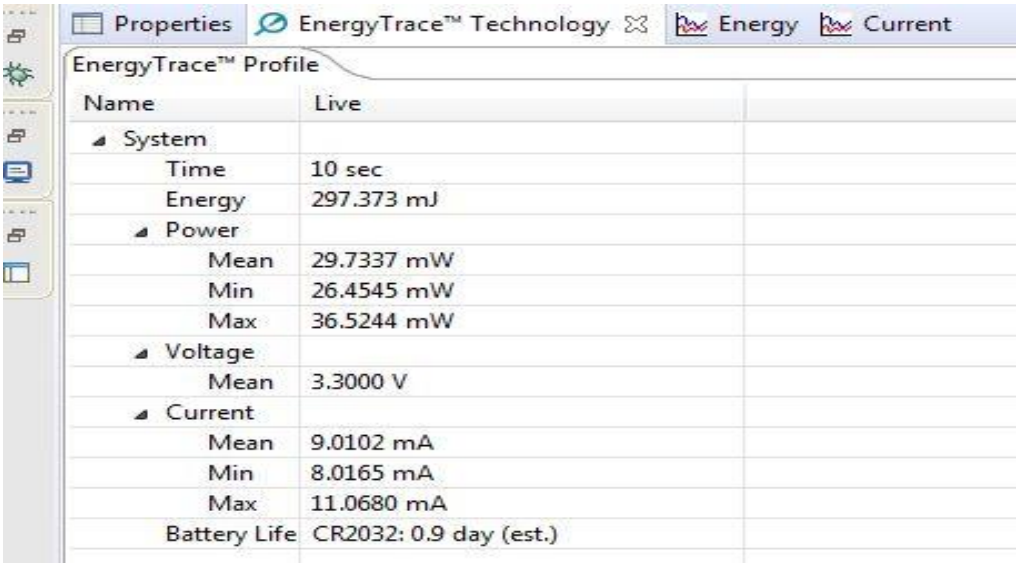
- Πρώτα Διαλέγουμε τον χρόνο δειγματοληψίας τον μετρήσεων από το εικονίδιο με το ρολόι

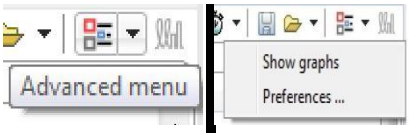
- Έπειτα ξεκινάμε την δειγματοληψία μετρήσεων από το εικονίδιο με το πράσινο τρίγωνο για τον χρόνο που επιλέχθηκε από το ρολόι π.χ. 10 sec



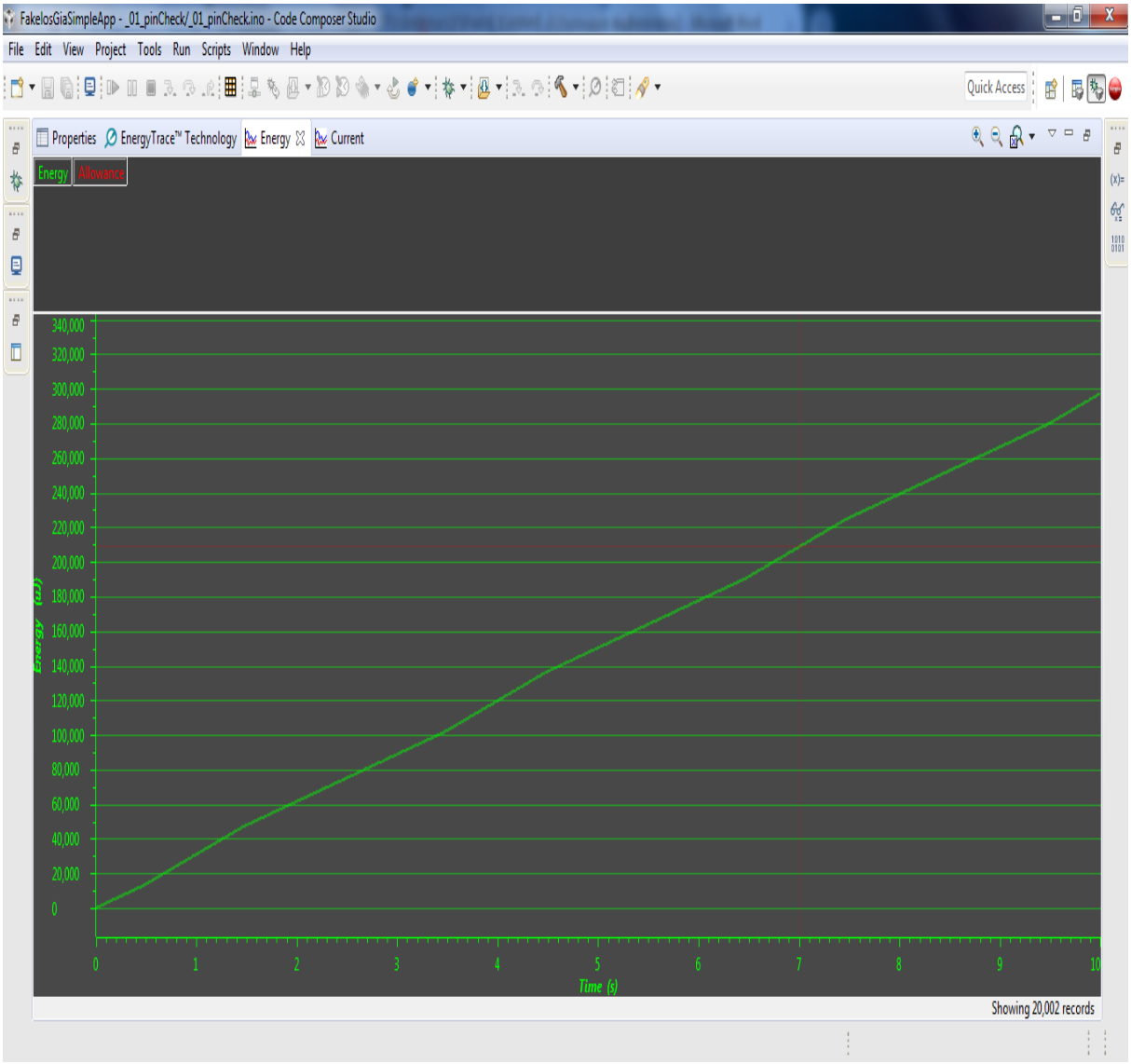
- Στην συνέχεια για να δούμε ξεκάθαρα της μετρήσεις, κάνουμε maximize το παράθυρο του Energy Trace από το κουμπί 

- Οι μετρήσεις παίρνονται σε πραγματικό χρόνο και σταματάει η δειγματοληψία έπειτα από τον χρόνο που ρυθμίσαμε να σταματήσει π.χ. 10 sec

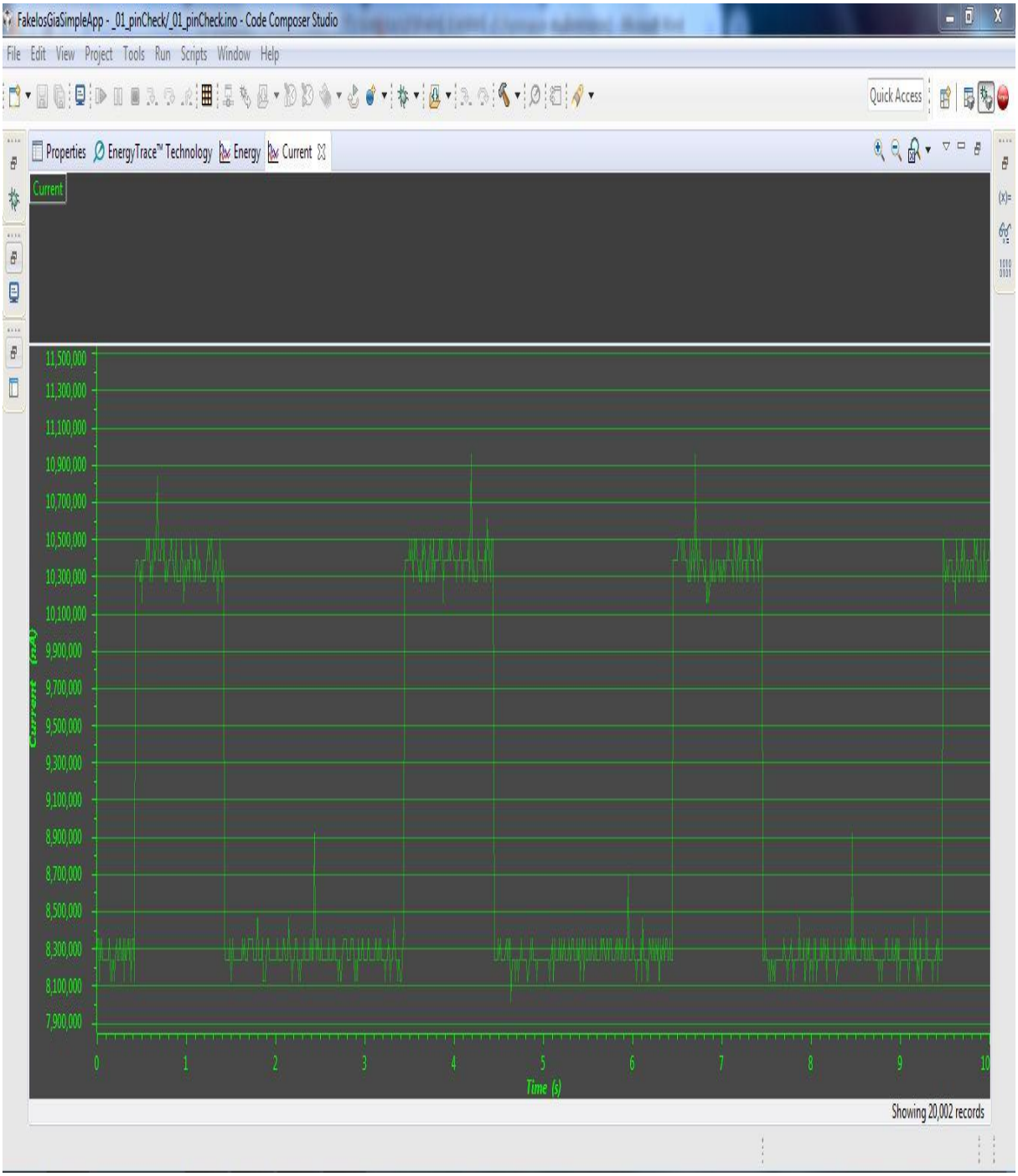




- Έπειτα από το κουμπί του Advanced Menu έχουμε πρόσβαση στα γραφήματα που δημιουργήθηκαν και στις ρυθμίσεις του Energy Trace που είδαμε παραπάνω. Πατώντας το Show graphs παίρνουμε τα γραφήματα: Ενέργειας συνάρτηση του χρόνου Energy (μJ) = f(t(sec)) που φαίνεται στην Εικόνα 32. και το γράφημα Ρεύματος συναρτήση του χρόνου Current (nA) = f(t(sec)) που φαίνεται στην Εικόνα 33. , επίσης πατώντας αριστερό κλικ του ποντικιού πατημένο πάνω στα γραφήματα εμφανίζεται ένας κόκκινος σταυρός ο οποίος μας βοηθάει να τα διαβάσουμε ποιο εύκολα, αν αφήσουμε το αριστερό κλικ ο σταυρός μένει πάνω στο γράφημα αλλά αχνοφαίνεται



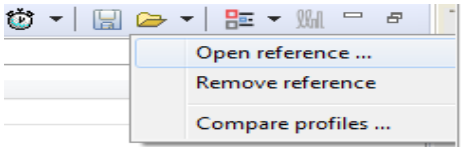
Εικόνα 33. Γράφημα ενέργειας συνάρτηση του χρόνου - Energy (μJ) = f(t(sec))

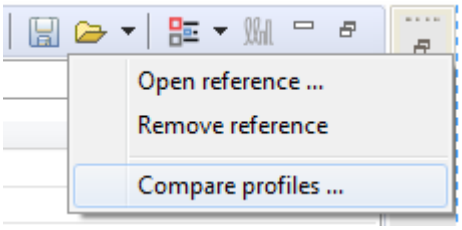


Εικόνα 34. Γράφημα ρεύματος συναρτήση του χρόνου Current (nA) = f(t(sec))

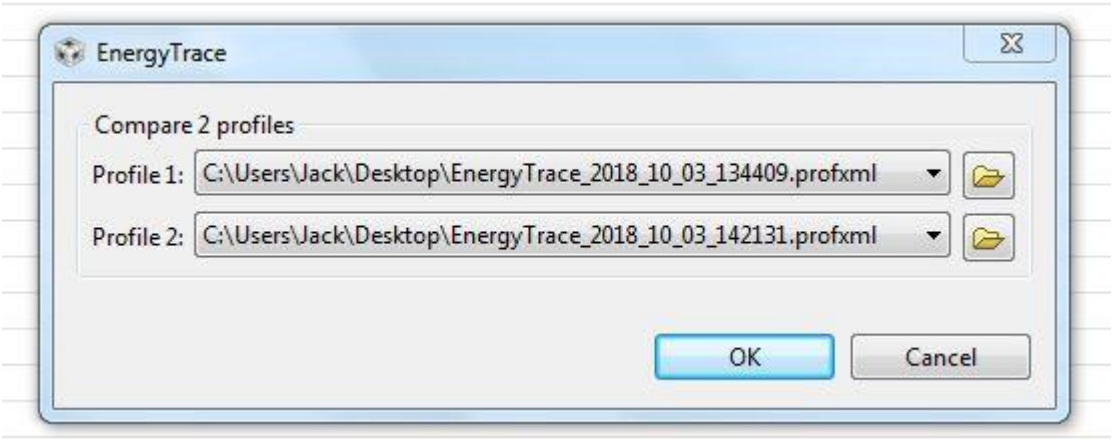
- Επίσης μπορούμε να αποθηκεύσουμε τις μετρήσεις μας, πατώντας το κουμπί της αποθήκευσης (Save current energy profile) 

, αποθηκεύεται σε μορφή .profxml

- Οποιαδήποτε στιγμή μπορούμε να ανοίξουμε της αποθηκευμένες μετρήσεις πατώντας το κουμπί ανοίγματος (Open Reference) 

και επιλέγουμε το αρχείο μετρήσεων που θέλουμε να ανοίξουμε
- Επίσης μπορούμε να συγκρίνουμε 2 αρχεία μεταξύ τους επιλέγοντας το κουμπί σύγκρισης (Compare profiles) 

επιλέγουμε τα 2 αρχεία και κάνει την σύγκριση δίνοντας την ποσοστιαία απόκλιση σε (%) το Delta(%)

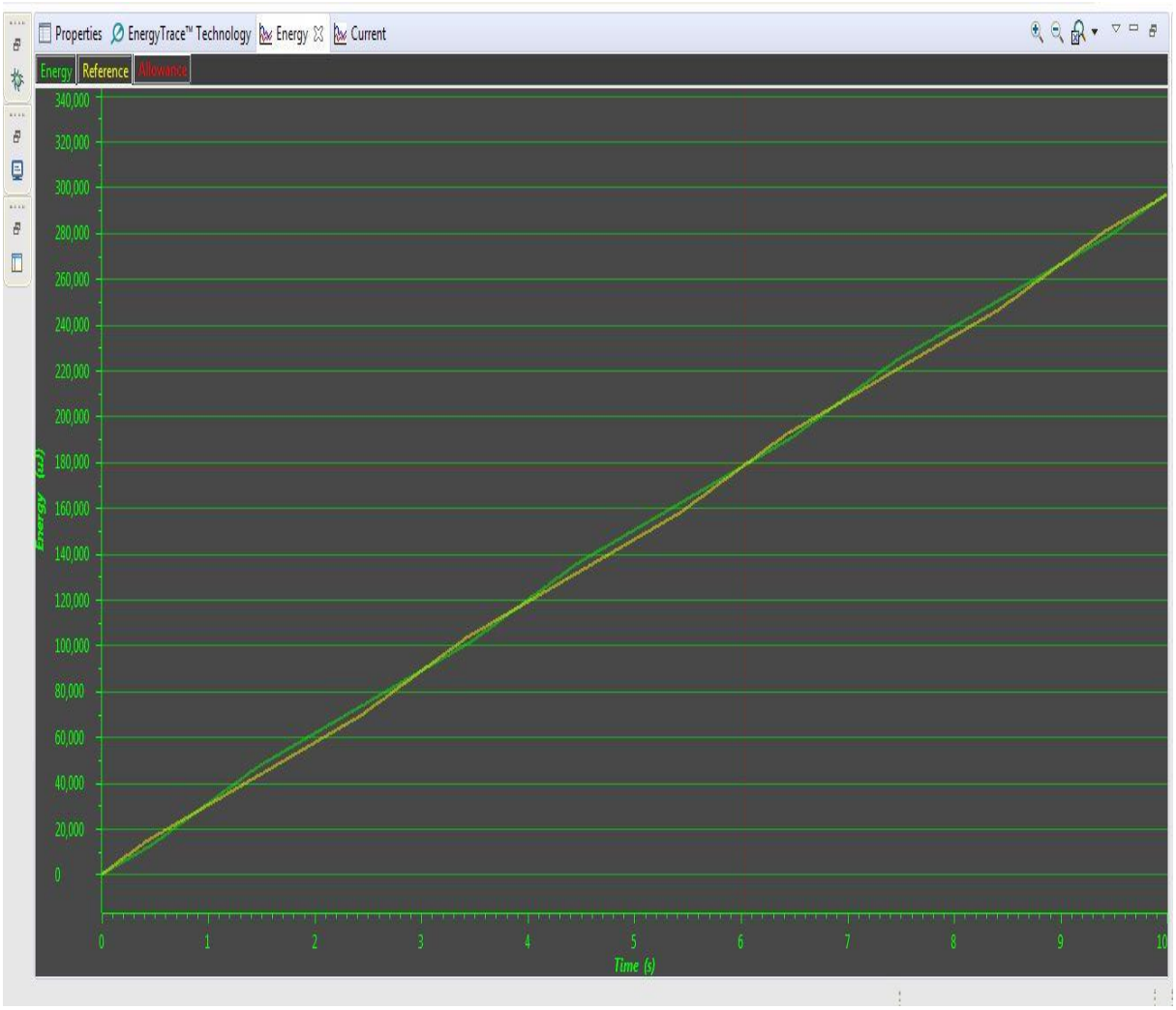


Properties | EnergyTrace™ Technology | Energy | Current

EnergyTrace_2018_10_03_134409 <-> EnergyTrace_2018_10_03_142131

EnergyTrace™ Profile

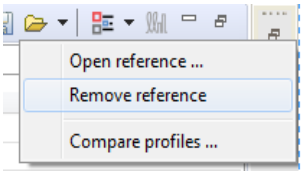
Name	Energy 1	Delta (%)	Energy 2
System			
Time	10 sec		10 sec
Energy	297.373 mJ	0.2	296.680 mJ
Power			
Mean	29.7337 mW	0.2	29.6657 mW
Min	26.4545 mW	0.0	26.4545 mW
Max	36.5244 mW	0.0	36.5244 mW
Voltage			
Mean	3.3000 V	0.0	3.3000 V
Current			
Mean	9.0102 mA	0.2	8.9896 mA
Min	8.0165 mA	0.0	8.0165 mA
Max	11.0680 mA	0.0	11.0680 mA
Battery Life	CR2032: 0.9 day (est.)	-0.2	CR2032: 0.9 day (est.)



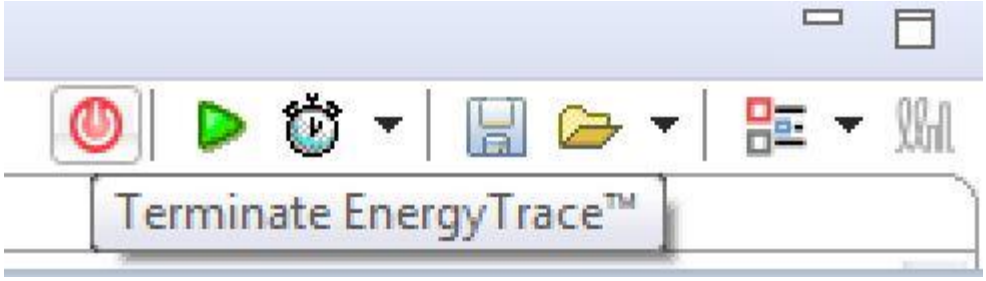
Εικόνα 35. Γράφημα σύγκρισης ενέργειας συνάρτηση του χρόνου των 2 αρχείων

με την πράσινη γραμμή βλέπουμε το πρώτο γράφημα και με την κίτρινη το δεύτερο.

- Έπειτα αν θέλουμε αφαιρούμε το δεύτερο αρχείο αναφοράς (Reference) γίνεται πατώντας το κουμπί αφαίρεσης αναφοράς (Remove reference)



- Τέλος πατάμε το κουμπί τερματισμού (Terminate EnergyTrace) του Energy Trace

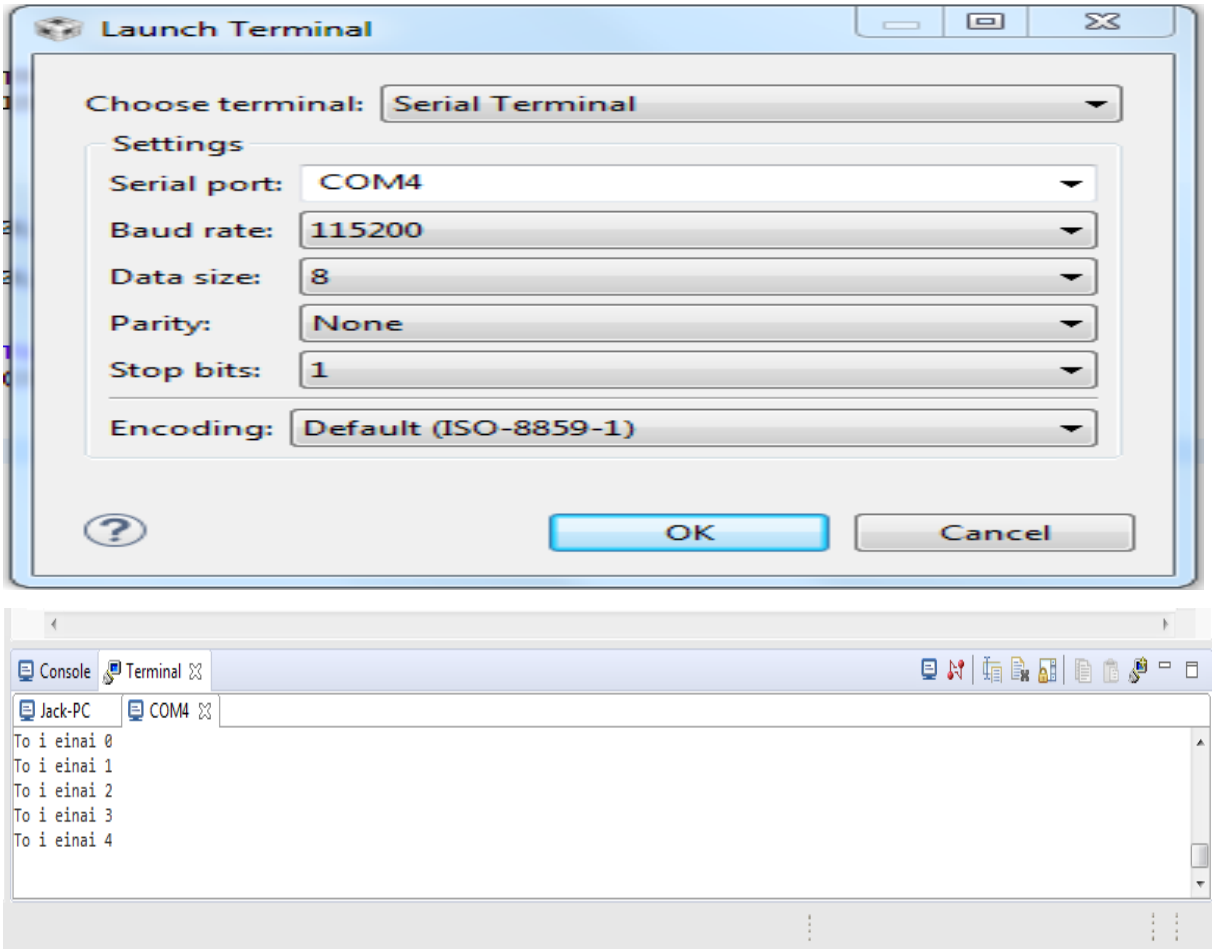


κλείνοντας το Energy Trace επιστρέφει αυτόματα στο CCS Edit Mode

Λειτουργία Serial Monitor

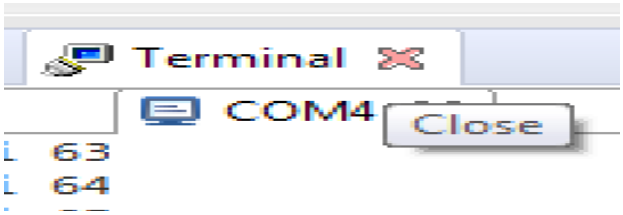
Για να ενεργοποιηθεί το Serial Monitor στο CCS IDE πατάμε στο εικονίδιο του Serial Monitor  και γίνονται οι βασικές ρυθμίσεις:

- διαλέγουμε που είναι συνδεμένη η συσκευή (Choose terminal), διαλέγουμε σειριακή θύρα (Serial port)
- βάζουμε ρυθμό μετάδοσης ίδιο με της συσκευής που συνδέσαμε (Baud rate) και πατάμε OK και εμφανίζεται το Serial Monitor κάτω από τον κώδικα του προγράμματος όπως φαίνεται στην Εικόνα 36.



Εικόνα 36. Serial Monitor

- Πατώντας το κόκκινο Χ δίπλα από την λέξη Terminal κλείνει το Serial Monitor

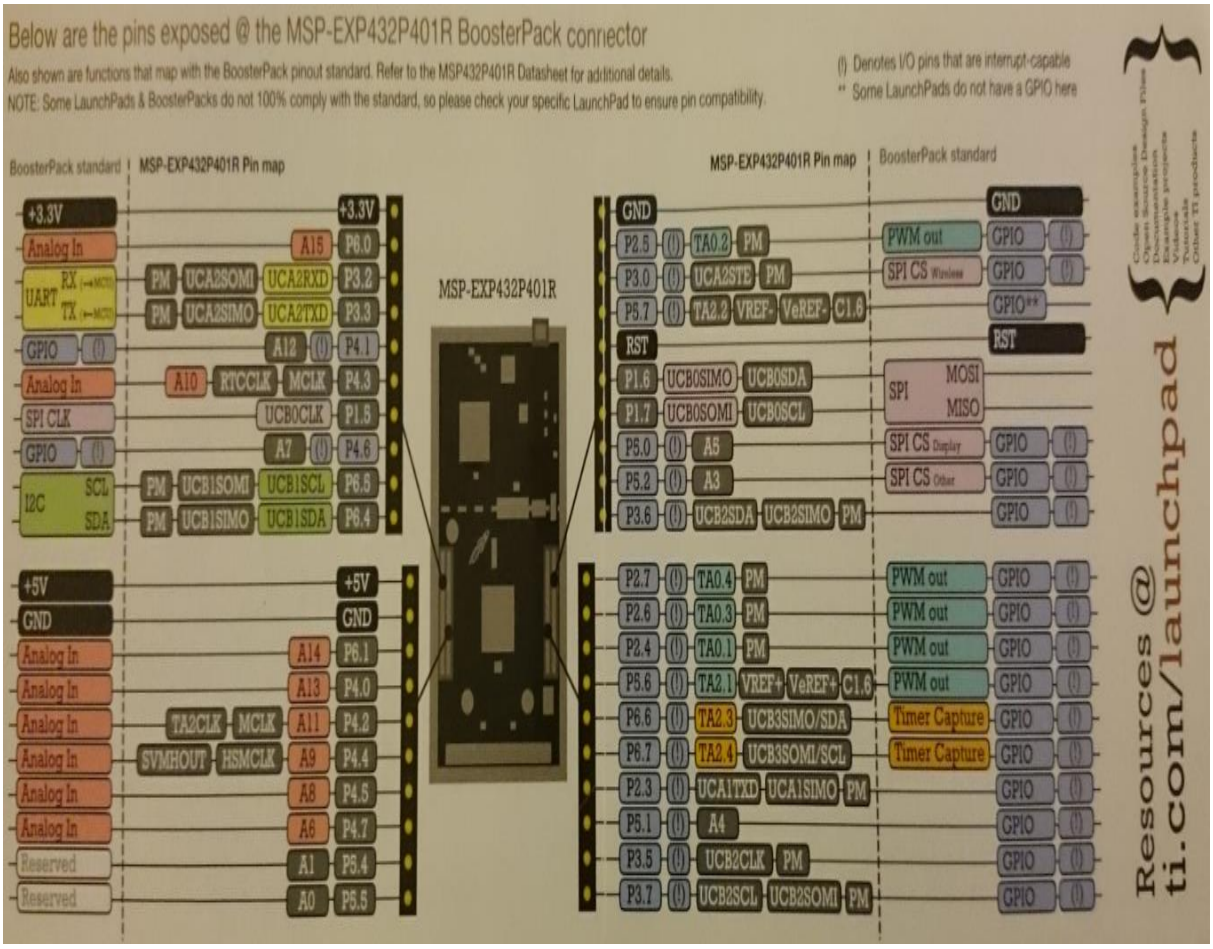


Ανάλυση βασικών καταχωρητών (Registers) του MSP432P401R

Τεχνική Μασκαρίσματος των Bits (Bitmask Technique)

Ο MSP432P401R διαθέτει 11 ψηφιακές θύρες (Digital Ports) από την P1 έως την P10 και την PJ εκ τον οποίων η κάθε μία θύρα έχει εύρος 8 γραμμές (8 bits).

Στο αναπτυξιακό MSP-EXP432P401R που χρησιμοποιήθηκε δεν χρησιμοποιήθηκαν όλες οι θύρες του MSP432P401R αλλά μόνο αυτές που βρισκόντουσαν στις θύρες συνδέσεων (J1 - J4) στο 40 pins Booster Pack Connector, που συνολικά είναι 33 εκμεταλλεύσιμες άμεσα I/O εκ το οποίον προέρχονται κυρίως από τις θύρες: P1, P2, P3, P4, P5 και P6 όπως φαίνεται αναλυτικά και στην Εικόνα 37 και Εικόνα 13.



Εικόνα 37. Οι 40 ακροδέκτες (Pins) του Booster Pack Connector με τις λειτουργίες τους

Όλες οι I/O είναι ψηφιακές και ο έλεγχος του γίνεται από τους καταχωρητές. Για να μπορέσουμε όμως να επικοινωνήσουμε με τους καταχωρητές δηλαδή να διαβάσουμε (Read) ή για να γράψουμε (Write) σε αυτούς χρησιμοποιείται η τεχνική μασκαρίσματος (Masking), δηλαδή χρησιμοποιείται μια μάσκα (Mask) που είναι συνήθως 8-bits για τον λόγο ότι και

θύρες είναι 8-bits (Υπάρχουν διαφορών μεγεθών μάσκες, όχι μόνο 8-bits) και μια ψηφιακή πράξη (bitwise operation) που μπορεί να είναι:

- AND με το σύμβολο &
- OR με το σύμβολο |
- XOR με το σύμβολο ^

Βασικά μασκαρίσματα για να γράψουμε (Write) στους καταχωρητές:

Μασκάρισμα με την πράξη bitwise AND (&) για να γράψουμε μηδέν σε μια θέση bit ενός καταχωρητή, μασκάρουμε με μηδενικό σε όποια θέση θέλουμε να την κάνουμε 0 και τα υπόλοιπα bits τα αφήνουμε απείρακτα βάζοντας 1 στην μάσκα, είναι βασισμένο στον πίνακα αληθείας της AND.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Εικόνα 38. Πίνακας Αληθείας της AND

Παράδειγμα σε έναν 8-bit καταχωρητή, αν ο P2OUT = 0b11111111 και θέλουμε το 7 bit και το 3 bit (από 0 έως 7, η αρίθμηση ξεκινάει από το 0) να γίνει 0 τότε μασκάρουμε με μάσκα AND ως εξής:

P2OUT &= 0b01110111; ή P2OUT = 0b01110111 & 0b11111111;

$$\begin{pmatrix} \text{Μάσκα} \\ \text{πράξη AND (\&)} \\ \text{Αποτέλεσμα} \end{pmatrix} \begin{matrix} 01110111 \\ 11111111 \\ 01110111 \end{matrix}$$

Μασκάρισμα με την πράξη bitwise OR (|) για να γράψουμε άσσο σε μια θέση bit ενός καταχωρητή, μασκάρουμε με άσσο σε όποια θέση θέλουμε να την κάνουμε 1 και τα υπόλοιπα bits τα αφήνουμε απείρακτα βάζοντας 0 στην μάσκα, είναι βασισμένο στον πίνακα αληθείας της OR.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Εικόνα 39. Πίνακας Αληθείας της OR

Παράδειγμα σε έναν 8-bit καταχωρητή, αν ο P2OUT = 0b00000000 και θέλουμε το 7 bit και το 3 bit να γίνει 1 τότε μασκάρουμε με μάσκα OR ως εξής:

P2OUT |= 0b10001000; ή P2OUT = 0b10001000 | 0b00000000;

$$\left(\begin{array}{c} Μάσκα \\ \hline πράξη\ OR\ (|) \\ \hline Αποτέλεσμα \end{array}\right) \begin{array}{c} 10001000 \\ \hline 00000000 \\ \hline 10001000 \end{array}$$

Μασκάρισμα με την πράξη bitwise XOR (^) για να εναλλάξουμε (Toggle) την κατάσταση μιας θέσης bit ενός καταχωρητή, από άσσο να την κάνουμε μηδέν και αντίστροφα, μασκάρουμε με άσσο σε όποια θέση θέλουμε να την κάνουμε εναλλαγή και τα υπόλοιπα bits τα αφήνουμε απείρακτα βάζοντας 0 στην μάσκα, είναι βασισμένο στον πίνακα αληθείας της XOR.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Εικόνα 40. Πίνακας Αληθείας της XOR

Παράδειγμα σε έναν 8-bit καταχωρητή, αν ο P2OUT = 0b11110000 και θέλουμε το 7 bit και το 3 bit να γίνει εναλλαγή κατάστασης τότε μασκάρουμε με μάσκα XOR ως εξής:

P2OUT ^= 0b10001000; ή P2OUT = 0b10001000 ^ 0b11110000;

$$\left(\begin{array}{c} Μάσκα \\ \hline πράξη\ XOR\ (\hat{}) \\ \hline Αποτέλεσμα \end{array}\right) \begin{array}{c} 10001000 \\ \hline 11110000 \\ \hline 01111000 \end{array}$$

Βασικό μασκάρισμα για να διαβάσουμε (read) την κατάσταση (status) ενός συγκεκριμένου bit στους καταχωρητές:

Μασκάρισμα με την πράξη bitwise AND (&) για να διαβάσουμε την κατάσταση ενός bit σε μια συγκεκριμένη θέση ενός καταχωρητή, μασκάρουμε με άσσο την θέση που θέλουμε να δούμε την κατάσταση της και τα υπόλοιπα bits τα βάζουμε μηδέν στην μάσκα, είναι βασισμένο στον πίνακα αληθείας της AND (Εικόνα 38.)

Παράδειγμα σε έναν 8-bit καταχωρητή, αν ο P2OUT = 0b10011001 και θέλουμε να δούμε την κατάσταση του 3 bit τότε μασκάρουμε με μάσκα AND ως εξής:

int Read = 0b00001000 & P2OUT;

$$\begin{pmatrix} \frac{Μάσκα}{πράξη\ AND\ (\&)} \\ Αποτέλεσμα \end{pmatrix} \begin{matrix} 00001000 \\ \hline 10011001 \\ \hline 00001000 \end{matrix}$$

Δεύτερη περίπτωση αν ο P2OUT = 0b10010001 τότε:

int Read= 0b00001000 & P2OUT;

$$\begin{pmatrix} \frac{Μάσκα}{πράξη\ AND\ (\&)} \\ Αποτέλεσμα \end{pmatrix} \begin{matrix} 00001000 \\ \hline 10010001 \\ \hline 00000000 \end{matrix}$$

Καταχωρητές ψηφιακών θυρών

Καταχωρητές διεύθυνσης [Direction Registers (PxDIR)]

Το κάθε bit στον PxDIR καταχωρητή, διαλέγει την διεύθυνση (αν είναι είσοδος ή έξοδος το Pin) του αντίστοιχου I/O pin αν είναι ρυθμισμένο για I/O λειτουργία. Συνήθως για τις περισσότερες περιπτώσεις περιφερειακών λειτουργιών (peripheral functions) ο PxDIR ελέγχει την διεύθυνση τον I/O. Τα bits του PxDIR που είναι επιλεγμένα για τις περιφερειακές λειτουργίες πρέπει να ρυθμιστούν όπως απαιτείται από τις περιφερειακές λειτουργίες. Για ορισμένες δευτερεύουσες λειτουργίες όπως το eUSCI (Enhanced Universal Serial Communication Interface), η διεύθυνση των I/O ελέγχεται απευθείας από την δευτερεύουσα λειτουργία και όχι από τον καταχωρητή.

Στο καταχωρητή PxDIR το γράμμα 'χ' αναφέρεται στην όνομα της θύρας π.χ. P2DIR άρα ρυθμίζουμε την διεύθυνση των bits της θύρας 2.

Ρύθμιση Bits:

- Αν το Bit = 0: Το Pin της θύρας ρυθμίζεται σαν είσοδος (input direction)
- Αν το Bit = 1: Το Pin της θύρας ρυθμίζεται σαν έξοδος (output direction)

π.χ. P2DIR |= 0b10000000; Στην θύρα 2 το 7 Pin ρυθμίζεται σαν έξοδος, άρα το Pin με το όνομα P2.7 είναι έξοδος.

Δεύτερη περίπτωση: P2DIR &=0b01111111; το Pin P2.7 είναι είσοδος.

PxDIR, Port X Direction Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 1. Καταχωρητής PxDIR

Πίνακας 3. Περιγραφή καταχωρητή PxDIR				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxDIR	Read και Write (RW) (rw, στο rw-0)	0h (-0, στο rw-0)	Port X direction 0b = H θύρα ρυθμίζεται σαν είσοδος 1b = H θύρα ρυθμίζεται σαν έξοδος

Καταχωρητές εξόδου [Output Registers (PxOUT)]

Το κάθε bit του καταχωρητή PxOUT είναι η τιμή εξόδου του αντίστοιχου I/O pin όταν είναι ρυθμισμένο για I/O λειτουργία, αν είναι ρυθμισμένο σαν έξοδος (output direction).

- Αν το Bit = 0: η έξοδος είναι Low (Output is low)
- Αν το Bit = 1: η έξοδος είναι High (Output is high)

π.χ. P2DIR |= 0b10000000;
P2OUT |= 0b10000000;

Το Pin 2.7 είναι έξοδος και είναι High (3.3 V στην περίπτωση του MSP432P401R)

Εάν το pin είναι ρυθμισμένο σε I/O λειτουργία, ρυθμισμένο σαν είσοδος (input direction) και οι αντιστάσεις (Resistors) pull-up ή pull-down είναι ενεργοποιημένες, το αντίστοιχο bit στον καταχωρητή PxOUT επιλέγει (Select) αν ενεργοποιηθεί η pull-up ή pull-down αντίσταση.

- Αν το Bit = 0: Το Pin είναι pulled down
- Αν το Bit = 1: Το Pin είναι pulled up

PxOUT, Port X Output Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Σχήμα 2. Καταχωρητής PxOUT

Πίνακας 4. Περιγραφή καταχωρητή PxOUT				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxOUT	RW	Απροσδιόριστο (Undefined)	Port X output Όταν η I/O έχει ρυθμιστεί σαν έξοδος: 0b = Η έξοδος είναι Low. 1b = Η έξοδος είναι high. Όταν η I/O έχει ρυθμιστεί σαν είσοδος και οι pullups / pulldowns αντιστάσεις έχουν ενεργοποιηθεί: 0b = Επιλέγεται η Pulldown αντίσταση 1b = Επιλέγεται η Pullup αντίσταση

Καταχωρητές ενεργοποιήσεις των Pull-up ή Pull-down αντιστάσεων [Pull-up or Pull-down Resistor Enable Registers (PxREN)]

Το κάθε bit στον καταχωρητή PxREN ενεργοποιεί ή απενεργοποιεί την αντίστοιχη Pull-up ή Pull-down αντίσταση στο αντίστοιχο I/O pin. Το bit επιλέγει αν υπάρχει αντίσταση Pull-up ή Pull-down ή όχι στο αντίστοιχο pin.

- Αν το Bit = 0: Η αντίσταση Pull-up ή Pull-down είναι απενεργοποιημένη
- Αν το Bit = 1: Η αντίσταση Pull-up ή Pull-down είναι ενεργοποιημένη

Συνοψίζοντας την χρήση των PxDIR, PxREN και PxOUT στον πίνακα 5.

Πίνακας 5. Ρυθμίσεις I/O			
PxDIR	PxREN	PxOUT	Ρύθμιση I/O
0	0	X	Είσοδος
0	1	0	Είσοδος με Pull-down αντίσταση
0	1	1	Είσοδος με Pull-up αντίσταση
1	X	X	Έξοδος

Το X στον πίνακα 5. σημαίνει ότι είναι αδιάφορος όρος.

PxREN, Port X Pullup or Pulldown Resistor Enable Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxREN							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 3. Καταχωρητής PxREN

Πίνακας 6. Περιγραφή καταχωρητή PxREN				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxREN	RW	0h	Port X ενεργοποίηση pullup ή pulldown αντίστασης 0b = Απενεργοποίηση Pullup ή pulldown 1b = Ενεργοποίηση Pullup ή pulldown

Καταχωρητές εισόδου [Input Registers (PxIN)]

Το κάθε bit στον καταχωρητή PxIN αντικατοπτρίζει την τιμή σήματος εισόδου στο αντίστοιχο I/O pin, όταν το pin είναι ρυθμισμένο σε λειτουργία I/O. Αυτός ο καταχωρητής είναι μόνο για διάβασμα (read only).

- Αν το Bit = 0: Η είσοδος είναι Low
- Αν το Bit = 1: Η είσοδος είναι High

ΣΗΜΕΙΩΣΗ: Γράφοντας στους read only PxIN καταχωρητές

Γράφοντας σε αυτούς του read only καταχωρητές έχει ως αποτέλεσμα την αυξημένη κατανάλωση ρεύματος όσο γίνεται η απόπειρα να γράψεις (Write) σε αυτούς.

PxIN, Port X Input Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxIN							
r	r	r	r	r	r	r	r

Σχήμα 4. Καταχωρητής PxIN

Πίνακας 7. Περιγραφή καταχωρητή PxIN				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxIN	R (read only)	Undefined	Port X είσοδος 0b = Η είσοδος είναι Low 1b = Η είσοδος είναι High

Καταχωρητές επιλογής δύναμης οδήγησης εξόδου [Output Drive Strength Selection Registers (PxDS)]

Υπάρχουν διαθέσιμοι 2 είδη τύπων I/O. Ένας με κανονική δύναμη οδήγησης (Regular drive strength) εξόδου και ένας με υψηλή δύναμη οδήγησης εξόδου (High drive strength). Οι περισσότερες I/O έχουν κανονική δύναμη οδήγησης και μόνο μερικές επιλεγμένες I/O (οι P2.0, P2.1, P2.2, P2.3) έχουν υψηλή δύναμη οδήγησης. Ο καταχωρητής PxDS χρησιμοποιείται για να επιλέξει την δύναμη οδήγησης στις I/O με υψηλή δύναμη οδήγησης.

- Αν το Bit = 0: Οι I/O υψηλής δύναμης οδήγησης ρυθμίζονται για κανονική ρύθμιση δύναμης οδήγησης
- Αν το Bit = 1: Οι I/O υψηλής δύναμης οδήγησης ρυθμίζονται για υψηλή δύναμη οδήγησης

Ο καταχωρητής PxDS δεν έχει κανένα αποτέλεσμα στις I/O με κανονική δύναμη οδήγησης.

PxDS, Port X Drive Strength Selection Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 ,10 or J).

7	6	5	4	3	2	1	0
PxDS							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 5. Καταχωρητής PxDS

Πίνακας 8. Περιγραφή καταχωρητή PxDS				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxDS	RW	0h	Port X επιλογή δύναμης οδήγησης των I/O. 0b = κανονική δύναμη οδήγησης 1b = υψηλή δύναμη οδήγησης

Καταχωρητές επιλογής λειτουργιών [Function Select Registers (PxSEL0, PxSEL1)]

Τα pins των θυρών (Port pins) συνήθως είναι πολυπλεγμένα (multiplexed) με άλλες λειτουργίες (functions) περιφερειακών μονάδων όπως φαίνεται στις Εικόνες 37 και 13. Το κάθε port pin χρησιμοποιεί 2 bits για να επιλέξει την λειτουργία του, σαν απλή θύρα I/O ή μία από τις 3 πιθανές λειτουργίες των περιφερειακών μονάδων. Ο πίνακας 9. δείχνει πως να γίνεται η επιλογή των διαφόρων λειτουργιών περιφερειακών μονάδων στο pin. Το κάθε bit στους καταχωρητές PxSEL χρησιμοποιείται για να επιλέξει την λειτουργία του pin, σαν θύρα γενικού σκοπού[General Purpose I/O(GPIO)] ή σαν λειτουργία περιφερειακών μονάδων.

Πίνακας 9. Επιλογή λειτουργίας των I/O pin

PxSEL1	PxSEL0	I/O Function
0	0	Επιλέγεται η λειτουργία General Purpose I/O
0	1	Επιλέγεται η πρώτη λειτουργία (Primary module function)
1	0	Επιλέγεται η δεύτερη λειτουργία (Secondary module function)
1	1	Επιλέγεται η τρίτη λειτουργία (Tertiary module function is selected)

Ρυθμίζοντας τα bits των καταχωρητών PxSEL1 ή PxSEL0 για λειτουργία μονάδας (module function) δεν ρυθμίζεται αυτόματα η διεύθυνση (direction) του pin. Άλλες περιφερειακές λειτουργίες μονάδος, μπορεί να χρειάζονται να ρυθμιστούν τα bits του καταχωρητή PxDIR, στην ανάλογη διεύθυνση που χρειάζεται η λειτουργία της μονάδος.

Όταν ένας ακροδέκτης θύρας (port pin) επιλέγεται σαν είσοδος στις περιφερειακές μονάδες, το σήμα εισόδου σε αυτές τις περιφερειακές μονάδες είναι μια μανδαλωμένη αναπαράσταση του σήματος στον ακροδέκτη της συσκευής. Όσο οι καταχωρητές PxSEL1 και PxSEL0 είναι διάφορο από το 00, το εσωτερικό σήμα εισόδου ακολουθεί το σήμα στον ακροδέκτη για όλες τις συνδεμένες μονάδες.

Ωστόσο, εάν οι καταχωρητές PxSEL1 και PxSEL0 είναι ίση με το 00, η είσοδος στα περιφερειακά διατηρούν την τιμή του σήματος εισόδου στον ακροδέκτη της συσκευής πριν τα bits των καταχωρητών PxSEL1 και PxSEL0 ήταν reset.

Επειδή τα bits των καταχωρητών PxSEL1 και PxSEL0 δεν είναι σε συνεχόμενες γειτονικές διευθύνσεις, δεν είναι δυνατή η αλλαγή και των δύο bits ταυτόχρονα.

Για παράδειγμα, μια εφαρμογή μπορεί να χρειάζεται να αλλάξουμε τον ακροδέκτη P1.0 από I/O γενικού σκοπού στην τρίτη λειτουργία του P1.0. Αρχικά, P1SEL1 = 00h και P1SEL0 = 00h. Για να αλλάξουμε την λειτουργία, είναι αναγκαίο να γράψουμε και στους ταυτόχρονα P1SEL1 = 01h και P1SEL0 = 01h. Αυτό δεν είναι δυνατό, χωρίς πρώτα να περάσει από μια ενδιάμεση ρύθμιση και αυτή η ρύθμιση μπορεί να μην είναι επιθυμητή από την άποψη της εφαρμογής.

Ο συμπληρωματικός καταχωρητής PxSELC μπορεί να χρησιμοποιηθεί για να διαχειριστεί τέτοιες περιπτώσεις. Ο καταχωρητής PxSELC πάντα διαβάζει 0. Το κάθε bit του καταχωρητή PxSELC συμπληρώνει τα αντίστοιχα σχετικά bit των καταχωρητών PxSEL1 και PxSEL0 ταυτόχρονα. Για παράδειγμα, αρχικά με τιμές P1SEL1 = 00h και P1SEL0 = 00h, γράφοντας P1SELC = 01h προκαλεί τους καταχωρητές να γραφτούν ταυτόχρονα σε P1SEL1 = 01h και P1SEL0 = 01h.

PxSEL0, Port X Function Selection Register 0 (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxSEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 6. Καταχωρητής PxSEL0

Πίνακας 10. Περιγραφή καταχωρητή PxSEL0				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxSEL0	RW	0h	Επιλογή λειτουργία θύρας. Το κάθε bit αντιστοιχεί σε ένα κανάλι στην Port X. Οι τιμές των bits των καταχωρητών PxSEL0 και PxSEL1 συνδυάζονται για να γίνει η επιλογή της ζητούμενης λειτουργίας. Βλέπε συνέχεια στον πίνακα 11.

PxSEL1, Port X Function Selection Register 1 (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxSEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 7. Καταχωρητής PxSEL1

Πίνακας 11. Περιγραφή καταχωρητή PxSEL1				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxSEL1	RW	0h	Συνέχεια του πίνακα 10. 00b = Επιλέγεται η λειτουργία General Purpose I/O. 01b = Επιλέγεται η πρώτη λειτουργία. 10b = Επιλέγεται η δεύτερη λειτουργία. 11b = Επιλέγεται η τρίτη λειτουργία.

PxSELC, Port X Complement Selection (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J).

7	6	5	4	3	2	1	0
PxSELC							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 8. Καταχωρητής PxSELC

Πίνακας 12. Περιγραφή καταχωρητή PxSELC				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxSELC	RW	0h	Συμπληρωματική λειτουργία επιλογής θύρας. Αλλάζει την τιμή το καταχωρητών PxSEL1 και PxSEL0 ταυτόχρονα. Διαβάζει πάντα 0.

Διακοπές θύρας (Port Interrupts)

Όλες οι σημαίες διακοπών (interrupt flags) των θυρών Px, για κάθε μια συγκεκριμένη θύρα έχουν προκαθορισμένη προτεραιότητα (Priority), με PxIFG.0 να είναι η μεγαλύτερη και συνδυάζονται για να τροφοδοτήσουν ένα και μόνο φορέα διακοπών (interrupt vector). Η μεγαλύτερη προτεραιότητα ενεργοποίησης διακοπής παράγει έναν αριθμό στον καταχωρητή PxIV. Αυτός ο αριθμός μπορεί να αξιολογηθεί ή να προστεθεί στον αθροιστή προγράμματος (program counter) ώστε αυτόματα να εισέλθει στην κατάλληλη ρουτίνα λογισμικού (software routine). Απενεργοποιώντας τα interrupts στις θύρες Px δεν επηρεάζει την τιμή (value) στον καταχωρητή PxIV. Ο καταχωρητής PxIV είναι μόνο μισής λέξης προσβάσιμος (half-word).

Το κάθε bit στον καταχωρητή PxIFG είναι μια interrupt flag για το αντίστοιχο I/O pin και η flag ενεργοποιείται όταν ένα επιλεγμένο σήμα εισόδου λαμβάνει χώρα στο pin. Όλες οι interrupt flags στον καταχωρητή PxIFG ζητούν μια διακοπή όταν το αντίστοιχο bit στον καταχωρητή PxIE έχει οριστεί (set). Το λογισμικό μπορεί επίσης να θέσει κάθε PxIFG flag, παρέχοντας έναν τρόπο να παράγεις διακοπή που ξεκινάει από το λογισμικό (software initiated interrupt).

- Αν το Bit = 0: Δεν εκκρεμεί καμία διακοπή (No interrupt is pending)
- Αν το Bit = 1: Εκκρεμεί μια διακοπή (An interrupt is pending)

Μόνο οι μεταβάσεις, όχι τα σταθερά επίπεδα, προκαλούν διακοπές. Εάν οποιαδήποτε PxIFG σημαία οριστεί κατά την διάρκεια Px ρουτίνας εξυπηρέτησης διακοπής [interrupt service routine (ISR)] ή μετά την εκτέλεση της, η ορισμένη PxIFG flag παράγει άλλο ένα interrupt. Αυτό εξασφαλίζει ότι κάθε μεταβατική περίοδος αναγνωρίζεται.

Σημείωση: PxIFG flags όταν γίνεται αλλαγή στους PxOUT, PxDIR, ή PxREN

Γράφοντας στους PxOUT, PxDIR, ή PxREN μπορεί να έχει ως αποτέλεσμα να ορίσει τις αντίστοιχες PxIFG flags.

Οποιαδήποτε πρόσβαση (διάβασμα ή γράψιμο) στον καταχωρητή PxIV αυτόματα επαναφέρει (reset) την υψηλότερη εκκρεμή σημαία διακοπής. Εάν άλλη interrupt flag οριστεί, άλλη μια διακοπή άμεσος παράγεται μετά την εξυπηρέτηση της αρχικής διακοπής. Για παράδειγμα, ας υποθέσουμε ότι P1IFG.0 έχει την μεγαλύτερη προτεραιότητα, εάν η P1IFG.0 και η P1IFG.2 σημαίες οριστούν (set), όταν η interrupt service routine έχει πρόσβαση στον καταχωρητή P1IV, η P1IFG.0 γίνεται reset αυτόματα, με την ολοκλήρωση της P1IFG.0 ISR, η P1IFG.2 παράγει άλλο ένα interrupt.

PxIV, Port X Interrupt Vector Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10).

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved			PxIV				
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Σχήμα 9. Καταχωρητής PxIV

Πίνακας 13. Περιγραφή καταχωρητή PxIV				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 5	Κατοχυρωμένο (Reserved)	R	0h	Reserved. Διαβάζοντας στο επιστρέφει 0h
7 - 0	PxIV	R	0h	Port x interrupt vector value **00h** = No interrupt pending **02h** = Interrupt Source: Port x.0 interrupt; Interrupt Flag: PxIFG.0; Interrupt Priority: **Highest** **04h** = Interrupt Source: Port x.1 interrupt; Interrupt Flag: PxIFG.1 **06h** = Interrupt Source: Port x.2 interrupt; Interrupt Flag: PxIFG.2 **08h** = Interrupt Source: Port x.3 interrupt; Interrupt Flag: PxIFG.3 **0Ah** = Interrupt Source: Port x.4 interrupt; Interrupt Flag: PxIFG.4 **0Ch** = Interrupt Source: Port x.5 interrupt; Interrupt Flag: PxIFG.5 **0Eh** = Interrupt Source: Port x.6 interrupt; Interrupt Flag: PxIFG.6 **10h** = Interrupt Source: Port x.7 interrupt; Interrupt Flag: PxIFG.7; Interrupt Priority: **Lowest**

PxIFG, Port X Interrupt Flag Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10).

7	6	5	4	3	2	1	0
PxIFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 10. Καταχωρητής PxIFG

Πίνακας 14. Περιγραφή καταχωρητή PxIFG				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxIFG	RW	0h	Port X interrupt flag 0b = Δεν εκκρεμεί καμία διακοπή (No interrupt is pending) 1b = Εκκρεμεί μια διακοπή (An interrupt is pending)

<u>Καταχωρητής επιλογής μετώπου μετάβασης διακοπής [Interrupt Edge Select Registers (PxIES)]</u>

Το κάθε bit στον καταχωρητή PxIES επιλέγει το αντίστοιχο μέτωπο μετάβασης διακοπής (interrupt edge) για το αντίστοιχο pin.

- Αν το Bit = 0: Η αντίστοιχη σημαία PxIFG ορίζεται (set) όταν γίνεται μετάβαση από χαμηλό σε υψηλό μέτωπο (low-to-high transition)
- Αν το Bit = 1: Η αντίστοιχη σημαία PxIFG ορίζεται (set) όταν γίνεται μετάβαση από υψηλό σε χαμηλό μέτωπο (high-to-low transition)

<u>Σημείωση: Γράφοντα στον καταχωρητή PxIES</u>

Γράφοντα στον PxIES για το κάθε αντίστοιχο I/O μπορεί να έχει ως αποτέλεσμα να ορίσει τις αντίστοιχες σημαίες διακοπής

PxIES	PxIN	PxIFG
0 → 1	0	May be set
0 → 1	1	Unchanged
1 → 0	0	Unchanged
1 → 0	1	May be set

PxIES, Port X Interrupt Edge Select Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10).

7	6	5	4	3	2	1	0
PxIES							
rw	rw	rw	rw	rw	rw	rw	rw

Σχήμα 11. Καταχωρητής PxIES

Πίνακας 15. Περιγραφή καταχωρητή PxIES				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxIES	RW	Undefined	Port X interrupt edge select 0b = PxIFG flag ορίζεται σε low to high transition. 1b = PxIFG flag ορίζεται σε high to low transition.

Καταχωρητής ενεργοποιήσης των διακοπών [Interrupt Enable Registers (PxIE)]

Το κάθε bit στον καταχωρητή PxIE ενεργοποιεί την σχετική σημαία στον καταχωρητή PxIFG

- Αν το Bit = 0: Η διακοπή απενεργοποιείται (the interrupt is disabled)
- Αν το Bit = 1: Η διακοπή ενεργοποιείται (the interrupt is enabled)

PxIE, Port X Interrupt Enable Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10).

7	6	5	4	3	2	1	0
PxIE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 12. Καταχωρητής PxIE

Πίνακας 16. Περιγραφή καταχωρητή PxIE				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
7 - 0	PxIE	RW	0h	Port X ενεργοποίηση διακοπής 0b = Η αντίστοιχη διακοπή θύρας απενεργοποιείται 1b = Η αντίστοιχη διακοπή θύρας ενεργοποιείται

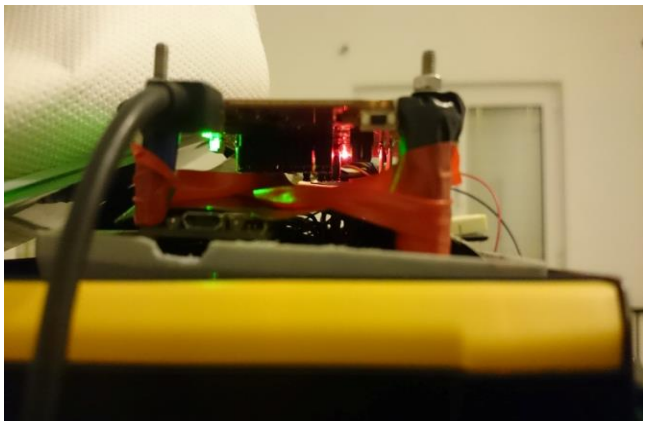
Ένα παράδειγμα με τους καταχωρητές και τις διακοπές στο CCS IDE, αναβοσβήνουμε το κόκκινο LED και όταν πατηθεί ο εσωτερικός διακόπτης SW1 ή SW2, τότε ενεργοποιείται η διακοπή και το κόκκινο LED σταματάει να αναβοσβήνει μένει στην κατάσταση που ήταν όταν ενεργοποιήθηκε η διακοπή και το πράσινο LED αναβοσβήνει 3 φορές έπειτα ξανά αναβοσβήνει το κόκκινο LED και συνεχίζει να αναβοσβήνει έως ότου ξανά πατηθεί ένας από τους 2 εσωτερικούς διακόπτες.

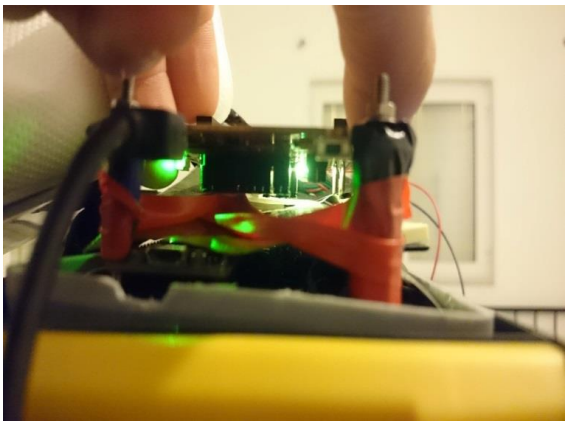
Κώδικας CCS σε γλώσσα προγραμματισμού C:

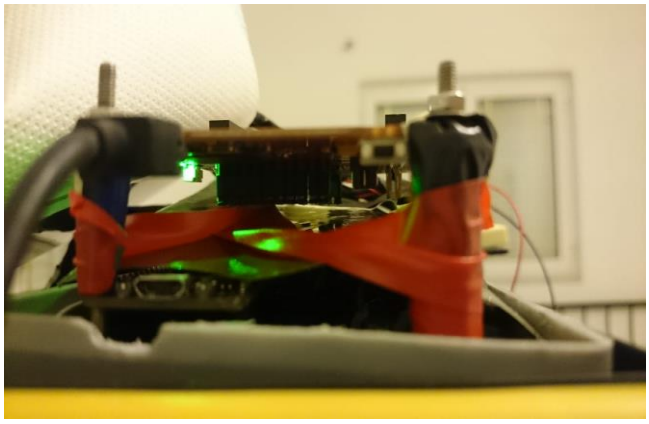
```
#include "msp.h"
void delayMs(int n);

void main(void) {
  WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD; // stop watchdog timer

  __disable_irq(); // genikh apenergopiohsh ton IRQs

  // ru8mizoume tis 8ures P1.1, P1.4 gia eisodos diakopton
  P1->SEL1 &= ~0x12; // ru8mizoume P1.1, P1.4 san aples I/O
  P1->SEL0 &= ~0x12;

  P1->DIR &= ~0x12; // P1.1, P1.4 orizontai san eisodoi

  P1->REN |= 0x12; // P1.1, P1.4 energopoihsei pull antistaseon
  P1->OUT |= 0x12; // Pull up epilegetai apo ton P1->OUT

  P1->IES |= 0x12; // to interrupt energopoietai sto katerxomeno metopo (high-to-
low transition)
  P1->IFG = 0; // ka8arizooume uparxon interrupt flags
  P1->IE |= 0x12; // energopoihsh ton interrupt gia ta P1.1, P1.4

  // ru8misei ton led sta P2.2-P2.0 gia to RGB LEDs
  P2->SEL1 &= ~7; // ru8misei ton P2.2-P2.0 san aples I/O
  P2->SEL0 &= ~7;
  P2->DIR |= 7; // P2.2-2.0 orizoume san e3odous
  P2->OUT &= ~7; // sbhnoume kai ta 3 led

  NVIC_SetPriority(PORT1_IRQn, 3); // 8etoume tin proteraiothta sto 3 ston NVIC
  NVIC_EnableIRQ(PORT1_IRQn); // energopoihsh ton interrupt ston NVIC

  __enable_irq(); // genikh energopoihsh ton IRQs

  //anabosbhnoume to RED LED (P2.0) sunexomena
  while (1) //atermon broxos
  {
    P2->OUT |= 0x01;
    delayMs(500);
    P2->OUT &= ~0x01;
    delayMs(500);
  }
}

//O diakopths SW1 einai sundemenos sto pin P1.1, o SW2 sto P1.4 kai oi 2
energopoioun to interrupt sto PORT1
```

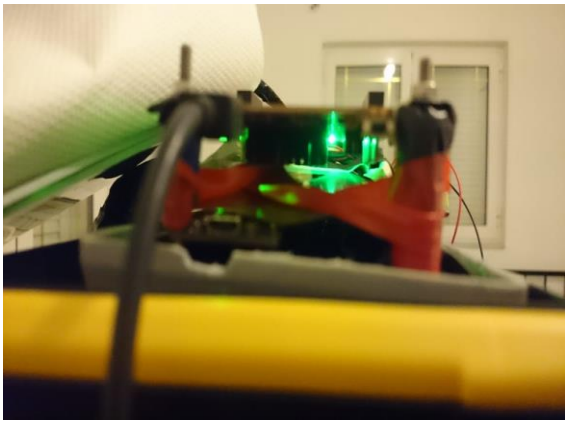
```
void PORT1_IRQHandler(void)
{
  int i;
  //Anabosbhnoume to green LED (P2.1) 3 fores
  for (i = 0; i < 3; i++)
  {
    P2->OUT |= 0x02;
    delayMs(500);
    P2->OUT &= ~0x02;
    delayMs(500);
  }
  P1->IFG &= ~0x12; // ka8arizoume to interrupt flag prin epistrepsoume
}

// delay n milliseconds (3 MHz CPU clock)
void delayMs(int n)
{
  int i, j;
  for (j = 0; j < n; j++)
    for (i = 750; i > 0; i--); //den kanei tipota gia 1 ms
}
```

Φωτογραφίες από την λειτουργία του προγράμματος, στις πρώτες 2 φωτογραφίες ανάβει και το κόκκινο LED ενώ στις επόμενες 2 είναι στην περίπτωση που είναι σβηστό το κόκκινο LED όταν ενεργοποιήθηκε η διακοπή.









Χρονιστές (Timers) και παραγωγή κυματομορφής PWM

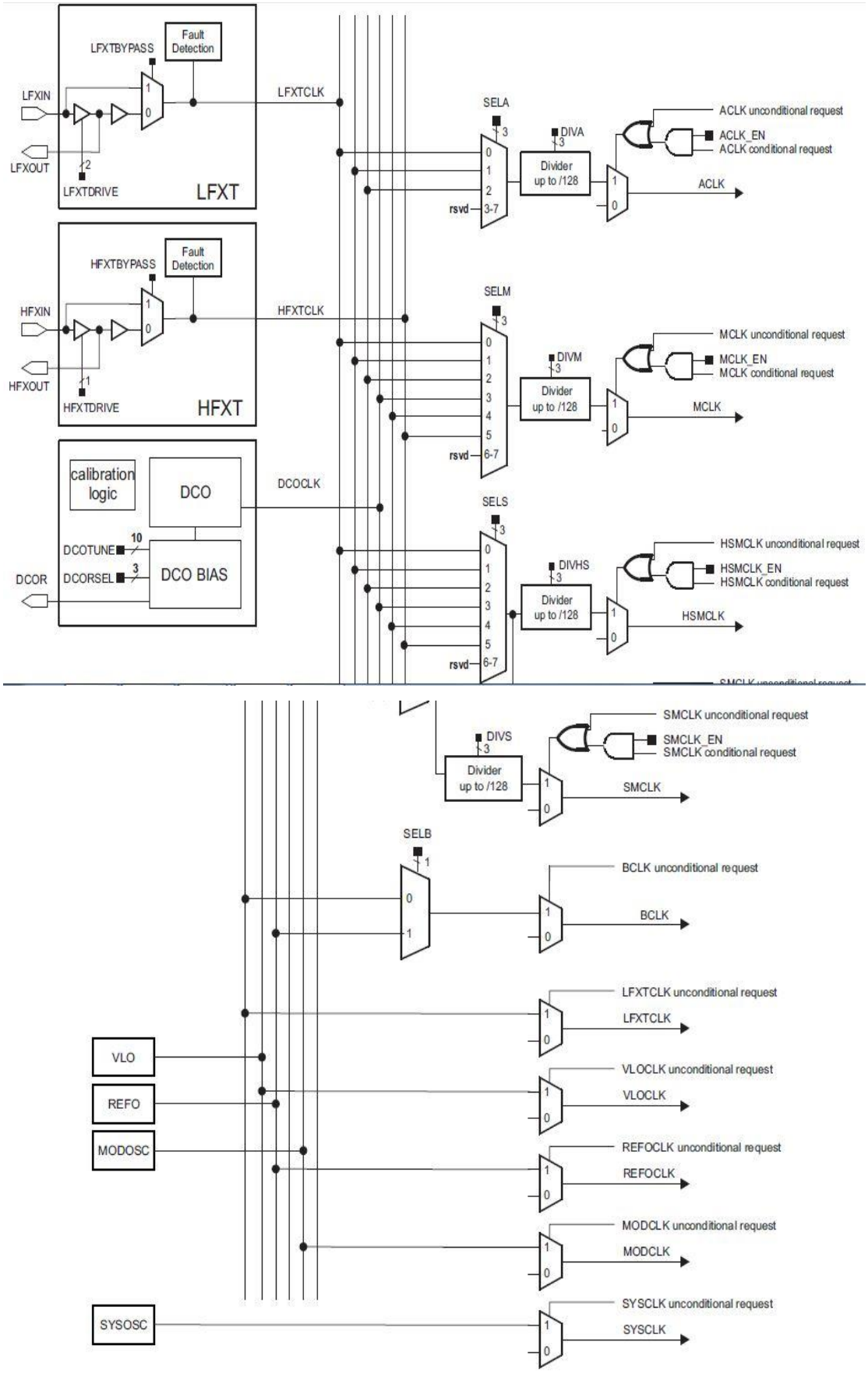
Ο MSP432P401R διαθέτει του χρονιστές: Timer32 και Timer_A, από τον Timer_A έχουμε την δυνατότητα παραγωγής κυματομορφής PWM.

Θα γίνει μια μικρή αναφορά στο ρολόι του συστήματος.

Πίνακας 17. Το Ρολόι του συστήματος (Clock System)	
Πηγή Ρολογιού [Ταλαντωτής (Oscillator)]	Συχνότητα λειτουργίας (Frequency)
LFXTCLK	32.768 Hz έως 32 KHz
HFXTCLK	1 MHz έως 48 MHz
DCOCLK	3 MHz [Προκαθορισμένο (default)]
VLOCLK	9.4 KHz
REFOCLK	32.768 KHz έως 128 KHz
MODCLK	25 MHz
SYSOSC	5 MHz

Οι έξοδοι του ρολογιού είναι:

Πίνακας 18. Έξοδοι ρολογιού	
Ονομασία εξόδου ρολογιού	Πηγές ρολογιού που χρησιμοποιεί
ACLK	LFXTCLK VLOCLK REFOCLK
MCLK	Και τις 7 πηγές του πίνακα 17.
HSMCLK	Όλες τις πηγές του πίνακα 17. εκτός την SYSOSC
SMCLK	Χρησιμοποίει σαν πηγή ρολογιού το HSMCLK αλλά έχει δικό του ξεχωριστό διαιρέτη (Divider) συχνότητας
BCLK	LFXTCLK REFOCLK SYSCLK



Εικόνα 41. Μπλοκ διάγραμμα του Ρολογιού του συστήματος (Clock System)

Χρονιστής Timer32:

Ο Timer32 αποτελείται από 2 προγραμματιζόμενους 32 bit ή 16 bit μετρητές προς τα κάτω (down counters) και μπορούν να παράγουν διακοπές όταν φτάνουν στο μηδέν.

Τα βασικά χαρακτηριστικά του Timer32 συμπεριλαμβάνουν:

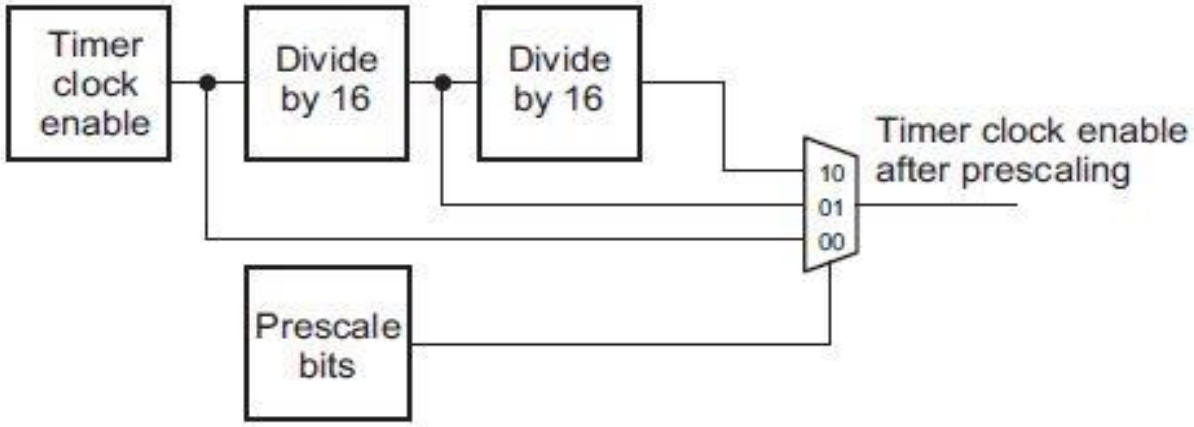
- Δύο ανεξάρτητους μετρητές ο καθένας ρυθμίζεται σαν 32-bit ή 16-bit μέγεθος μετρητή
- Τρείς διαφορετικές λειτουργιές χρονιστή (timer modes) υποστηριζόμενα για κάθε μετρητή
- Η μονάδα προβαθμίδωσης (Prescale unit) που διαιρεί το ρολόι εισόδου δια 1, 16 ή 256 φορές
- Ανεξάρτητες διακοπές από κάθε μετρητή, καθώς και συνδυαστικές διακοπές και από τους δύο μετρητές

Δύο ανεξάρτητοι χρονιστές (timers) είναι διαθέσιμοι στον Timer32. Για κάθε timer, οι παρακάτω λειτουργίες είναι διαθέσιμες:

- Λειτουργία ελεύθερου τρεξίματος (Free running mode): Ο μετρητής αναδιπλώνεται (wraps) αφού φτάσει στη μηδενική του τιμή και συνεχίζει να μετρά προς τα κάτω από την μέγιστη τιμή του. Αυτή είναι η προεπιλεγμένη λειτουργία.
- Περιοδική λειτουργία χρονιστή (Periodic timer mode): Ο μετρητής παράγει μια διακοπή σε ένα σταθερό διάστημα, επαναφορτώνοντας την αρχική τιμή μετά την αναδίπλωση του στο μηδέν.
- Λειτουργία μιας χρήσης χρονιστή (One-shot timer mode): Ο μετρητής παράγει μια διακοπή, όταν φτάσει στο μηδέν, έπειτα σταματάει να λειτουργεί μέχρι να επαναπρογραμματιστεί από το χρήστη. Αυτό μπορεί να επιτευχθεί είτε καθαρίζοντας το bit στο πεδίο One Shot Count bit στον καταχωρητή ελέγχου (control register), στην οποία περίπτωση η μέτρηση προχωρά σύμφωνα με την επιλογή του ελεύθερου τρεξίματος ή περιοδικού τρόπου λειτουργίας, ή με την εγγραφή μιας νέας τιμής στον καταχωρητή τιμής φόρτωσης (Load Value register).

Η ενεργοποίηση του ρολογιού του χρονιστή παράγεται από το Prescale unit. Η ενεργοποίηση χρησιμοποιείται στη συνέχεια από τον μετρητή για να δημιουργήσει ένα ρολόι με χρονισμό ενός από τα παρακάτω:

- Master Clock (MCLK)
- MCLK διαιρεμένο δια 16,παράγεται από 4 bits του prescale
- MCLK διαιρεμένο δια 256, παράγεται από συνολικά 8 bits του prescale



Εικόνα 42. Ενεργοποίηση διαίρεσης συχνότητας ρολογιού

Ο χρονιστής 1 και 2 έχουν πανομοιότυπους καταχωρητές άρα θα αναφερθούμε για τους βασικούς κατωρητές του χρονιστή 1. Καταχωρητής φόρτισης του χρονιστή 1, [Timer1 Load Register (T32LOAD1)]

(offset = 00h) [reset = 0h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD																															
rw-0																															

Σχήμα 13. Καταχωρητής T32LOAD1

Πίνακας 19. Περιγραφή καταχωρητή T32LOAD1				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
31 - 0	Φόρτιση (LOAD)	RW	0h	Είναι η τιμή από την οποία μειώνεται ο μετρητής του χρονιστή 1

Καταχωρητής τιμής του χρονιστή 1, T32VALUE1 Register

(offset = 04h) [reset = FFFFFFFFh]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE																															
r-FFFFFFFFh																															

Σχήμα 14. Καταχωρητής T32VALUE1

Αυτός ο καταχωρητής εμπεριέχει την τρέχων τιμή του Timer 1.

Πίνακας 20. Περιγραφή καταχωρητή T32VALUE1				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
31 - 0	Τιμή (VALUE)	R	FFFFFFFFh	Αναφέρει την τρέχουσα τιμή του μετρητή μείωσης (decrementing counter)

Καταχωρητής έλεγχου του χρονιστή 1, [Timer 1 Timer Control Register (T32CONTROL1)] (offset = 08h) [reset = 20h]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ENABLE	MODE	IE	Reserved	PRESCALE		SIZE	ONESHOT
r-0								rw-0	rw-0	rw-1	r-0	rw-0		rw-0	rw-0

Σχήμα 15. Καταχωρητής T32CONTROL1

Πίνακας 21. Περιγραφή καταχωρητή T32CONTROL1				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
31 - 8	Reserved	R	0h	Reserved
7	ENABLE	RW	0h	Enable bit 0b = O Timer απενεργοποιείτε (disabled) 1b = O Timer ενεργοποιείτε (enabled)
6	MODE	RW	0h	Mode bit 0b = O Timer είναι σε λειτουργία free-running 1b = O Timer είναι σε περιοδική λειτουργία (periodic)
5	IE	RW	1h	Interrupt enable bit 0b = Timer interrupt

				disabled 1b = Timer interrupt enabled
4	Reserved	R	0h	Reserved
3 - 2	PRESCALE	RW	0h	Prescale bits 00b = 0 στάδια του prescale, διαίρεση ρολογιού δια 1 01b = 4 στάδια του prescale, διαίρεση ρολογιού δια 16 10b = 8 στάδια του prescale, διαίρεση ρολογιού δια 256 11b = Reserved
1	SIZE	RW	Oh	Επιλογή λειτουργίας του μετρητή σε 16 ή 32 bit 0b = 16-bit counter 1b = 32-bit counter
0	ONESHOT	RW	0h	Επιλογή λειτουργίας μετρητή σε one-shot ή wrapping 0b = wrapping mode 1b = one-shot mode

Ένα παράδειγμα του Timer32 1 και 2, αναβοσβήνουμε το RGB LED του MSP-EXP432P401R σε διάφορους χρόνους. Ο μαθηματικός τύπος υπολογισμού του χρόνου του Timer32 είναι: $t = \frac{1}{\left(\frac{f_{clock}}{prescaler}\right)} \times T32LOAD1$ ή 2, σύμφωνα με αυτόν τον τύπο υπολογίζουμε τον χρόνο που αλλάζουν κατάσταση τα LEDs, το κόκκινο LED σε:

$$t_{red} = \frac{1}{\left(\frac{f_{clock}}{prescaler}\right)} \times T32LOAD1 = \frac{1}{\left(\frac{3\times10^6 Hz}{1}\right)} \times (2.3 \times 10^6) = 0.759\ Seconds$$

Το πράσινο LED σε:

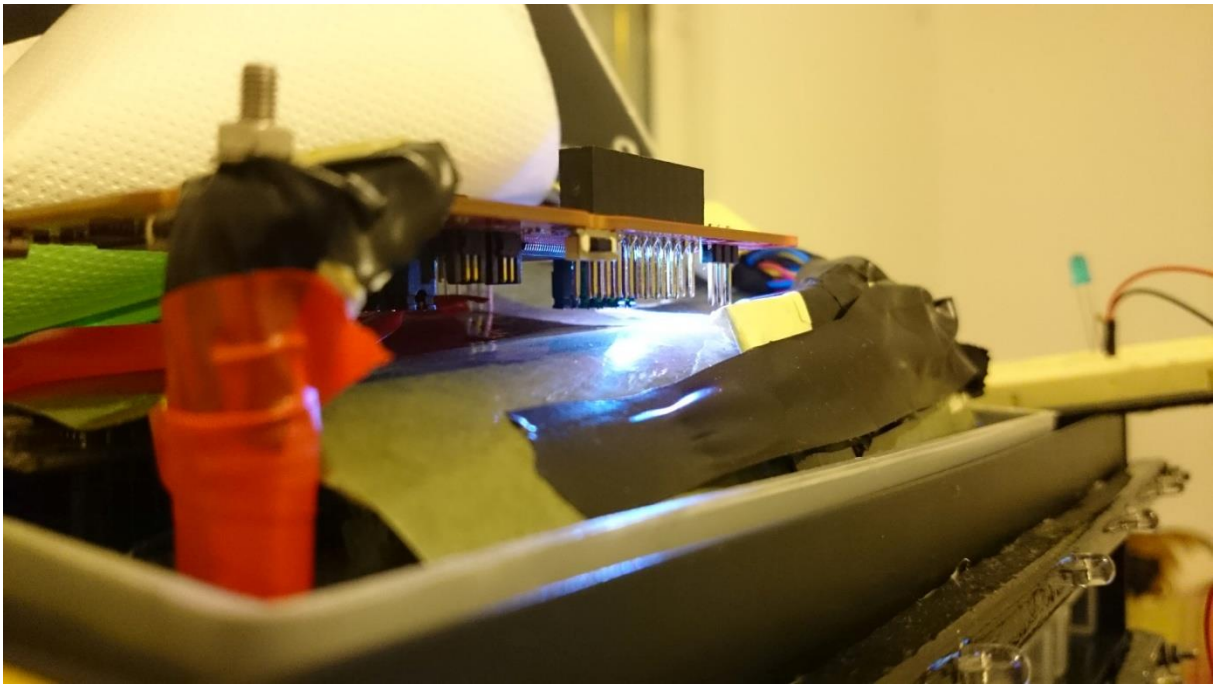
$$t_{green} = \frac{1}{\left(\frac{f_{clock}}{prescaler}\right)} \times T32LOAD2 = \frac{1}{\left(\frac{3 \times 10^6 Hz}{1}\right)} \times (1.9 \times 10^6) = 0.627\ Seconds$$

Το μπλέ LED σε 1 Second.

Κώδικας CCS σε γλώσσα προγραμματισμού C:

```
#include "msp.h"
void delayMs(int n);

int main(void)
{
  __disable_irq();

  //Arxikopoihsei tou RGB LED P2.2-P2.0
  P2->SEL1 &= ~7; // Ru8mish gia aplh I/O P2.2-P2.0
  P2->SEL0 &= ~7;
  P2->DIR |= 7; // P2.2-P2.0 orismos san e3odos

  //Ru8mish tou Timer32.1
  TIMER32_1->CONTROL = 0xC2; //xwris prescaler, periodic mode, 32-bit timer
  TIMER32_1->LOAD = 2300000 - 1; //orizoume tin timh tou reload
  TIMER32_1->CONTROL |= 0x20; // energopoihsh toy interrupt
  NVIC_SetPriority(T32_INT1_IRQn, 3); //8etoume proteraiothta sto 3 ston NVIC
  NVIC_EnableIRQ(T32_INT1_IRQn); //energopoihsh diakopwn ston NVIC

  //Ru8mish tou  Timer32.2
  TIMER32_2->CONTROL = 0xC2; //xwris prescaler, periodic mode, 32-bit timer
  TIMER32_2->LOAD = 1900000 - 1; //orizoume tin timh tou reload
  TIMER32_2->CONTROL |= 0x20; //energopoihsh toy interrupt
  NVIC_SetPriority(T32_INT2_IRQn, 4); //8etoume proteraiothta sto 4 ston NVIC
  NVIC_EnableIRQ(T32_INT2_IRQn); //energopoihsh diakopwn ston NVIC

  __enable_irq(); //genikh energopoihsh ton IRQs
  while (1)//atermon broxos
  {
    P2->OUT ^= 4; //anabosbhnei to blue LED ana 1 Second
    delayMs(1000);
  }
}

void T32_INT1_IRQHandler(void) //ana 0.759 Seconds
{
  TIMER32_1->INTCLR = 0; //clear raw interrupt flag
  P2->OUT ^= 1; //anabosbhnei to red LED
}

void T32_INT2_IRQHandler(void) //ana 0.627 Seconds
{
  TIMER32_2->INTCLR = 0; //clear raw interrupt flag
  P2->OUT ^= 2; //anabosbhnei to green LED
}

// delay milliseconds when system clock is at 3 MHz
void delayMs(int n)
{
  int i, j;
  for (j = 0; j < n; j++)
    for (i = 750; i > 0; i--); //ka8usterhsh 1 ms
}
```

Φωτογραφία από την λειτουργία του προγράμματος αναμμένα και τα 3 LED του RGB

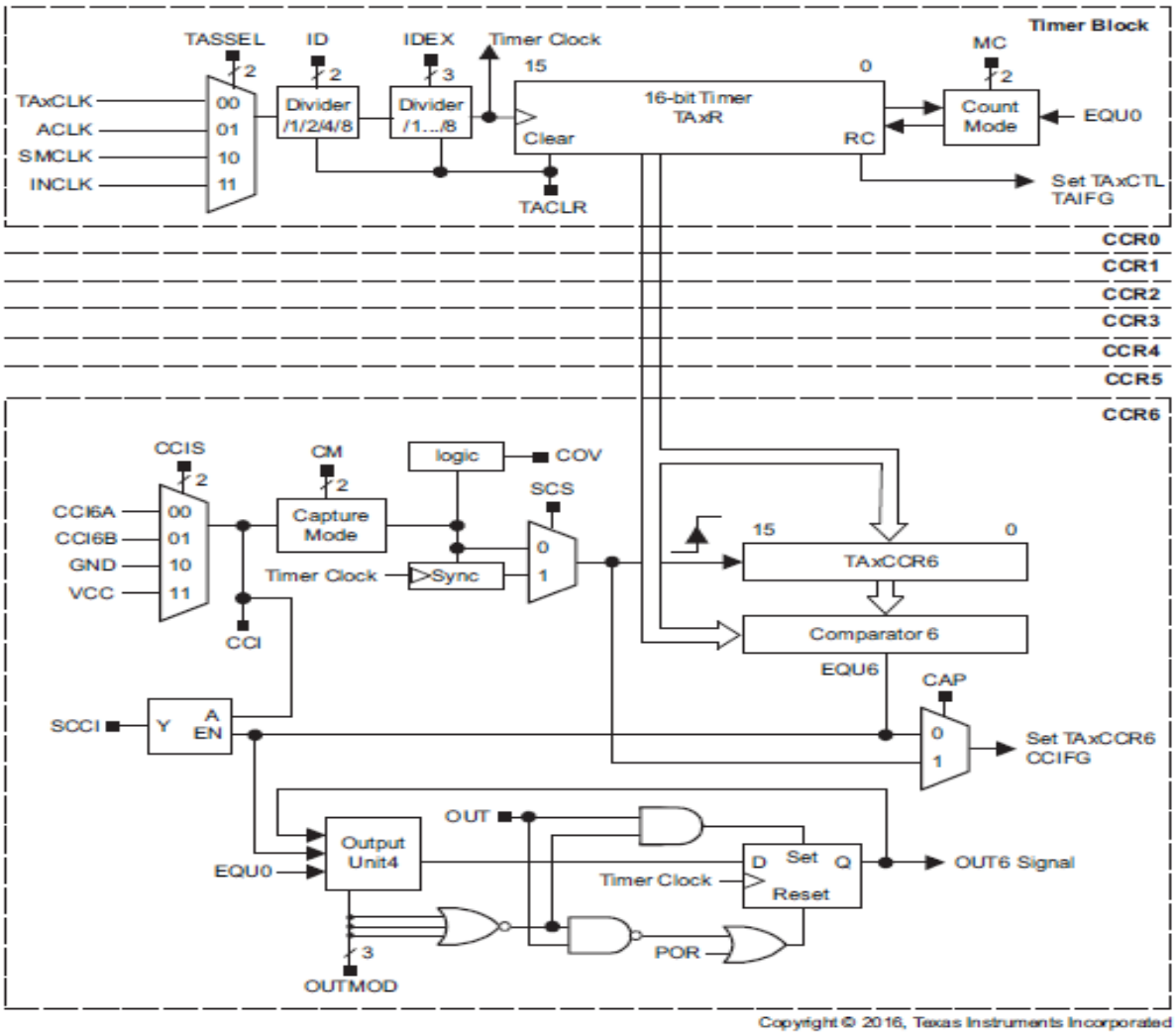


Χρονιστής Timer_A

Ο Timer_A είναι 16 bit χρονιστής/μετρητής με έως 7 καταχωρητές σύλληψης / σύγκρισης (capture / compare). Ο Timer_A μπορεί να υποστηρίξει πολλαπλά capture/compares, εξόδους PWM και χρονικά διαστήματα (interval timing). Ο Timer_A επί πλέον έχει εκτεταμένες δυνατότητες διακοπής. Οι διακοπές μπορούν να δημιουργηθούν από τον μετρητή σε συνθήκες υπερχείλισης (overflow) και από κάθε έναν από τους καταχωρητές σύλληψης / σύγκρισης.

Χαρακτηριστικά που περιλαμβάνονται στον Timer_A:

- Ασύγχρονος 16-bit timer/counter με 4 τρόπους λειτουργίας
- Δυνατότητα επιλογής και ρύθμισης πηγής ρολογιού
- Έως 7 ρυθμιζόμενους capture/compare καταχωρητές
- Ρυθμιζόμενες εξόδους με δυνατότητα διαμόρφωσης εύρους παλμού (PWM)
- Ασύγχρονη μανδάλωση εισόδου και εξόδου



Εικόνα 43. Μπλοκ διάγραμμα του Timer_A

Ο 16 bit καταχωρητής χρονιστής / μετρητής (timer/counter register), TAxR, οι αυξήσεις ή οι μειώσεις (εξαρτάται από τον τρόπο λειτουργίας) του γίνονται με κάθε ανερχόμενο μέτωπο του σήματος ρολογιού. Ο καταχωρητής TAxR μπορεί να διαβαστεί ή να γραφτεί με το λογισμικό. Επιπρόσθετα, ο χρονιστής μπορεί να παράγει διακοπές όταν υπερχειλίζει (overflows) δηλαδή όταν φτάνει την επιθυμητή τιμή και επιστρέφει στην αρχική τιμή που ξεκίνησε.

Ο καταχωρητής TAxR μπορεί να καθαριστεί ορίζοντας το TACLR bit. Ορίζοντας το TACLR καθαρίζει επίσης το διαιρέτη του ρολογιού και την διεύθυνση μέτρησης για την πάνω/κάτω λειτουργία (up/down mode).

Η επιλογή πηγής ρολογιού και διαιρέτη του χρονιστή Timer_A γίνεται από το ACLK, SMCLK, ή εξωτερικά από το TAxCLK ή INCLK. Η πηγή του ρολογιού επιλέγεται από τα

TASSEL bits. Η επιλεγμένη πηγή ρολογιού μπορεί να περάσει απευθείας στον χρονιστή η ναι διαιρεθεί (divided) δια 2,4, ή 8, χρησιμοποιώντας τα ID bits. Η επιλεγμένη πηγή ρολογιού μπορεί να υποστεί περισσότερη διαίρεση δια 2, 3, 4, 5, 6, 7, ή 8 χρησιμοποιώντας τα TAIDEX bits. Η λογική του διαιρέτη ρολογιού επαναφέρεται (reset) όταν ορίζεται (set) το TACLR.

Ο χρονιστής ξεκινάει να μετράει όταν πάρει παλμό ρολογιού.

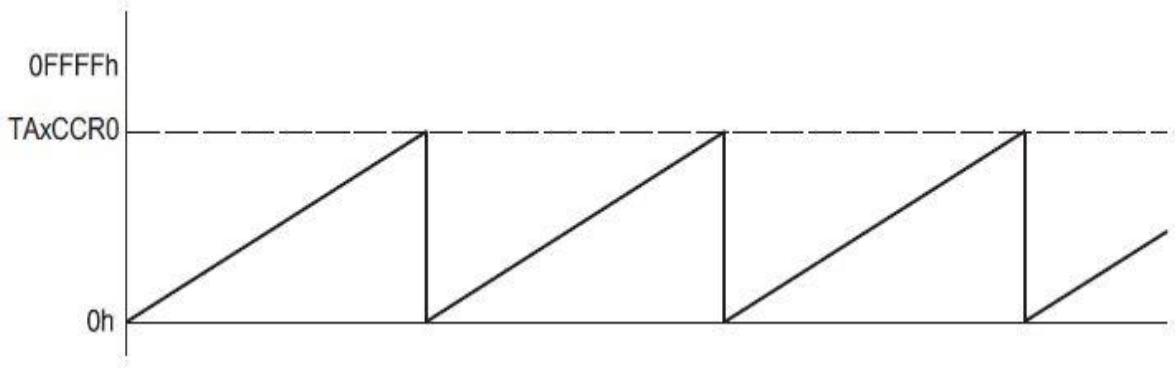
Έλεγχος λειτουργιών του Timer, ο Timer έχει 4 τρόπους λειτουργίας με της ονομασίες: σταματημένος (stop), πάνω(up), συνέχεις (continuous) και πάνω/κάτω (up/down). Οι τρόποι λειτουργίας επιλέγονται από τα MC bits.

Πίνακας 22. Τρόποι λειτουργίας χρονιστή (Timer Modes)

MC	Mode	Περιγραφή
00	Stop	Ο Timer είναι σταματημένος
01	Up	Ο Timer μετράει επανειλημμένα από το 0 έως την τιμή του TAxCCR0
10	Continuous	Ο Timer μετράει επανειλημμένα από το 0 έως το 0FFFFh
11	Up / Down	Ο Timer μετράει επανειλημμένα από το 0 έως την τιμή του TAxCCR0 και προς τα πίσω μέχρι να ξαναπάει στο 0

Λειτουργία προς τα πάνω (Up Mode):

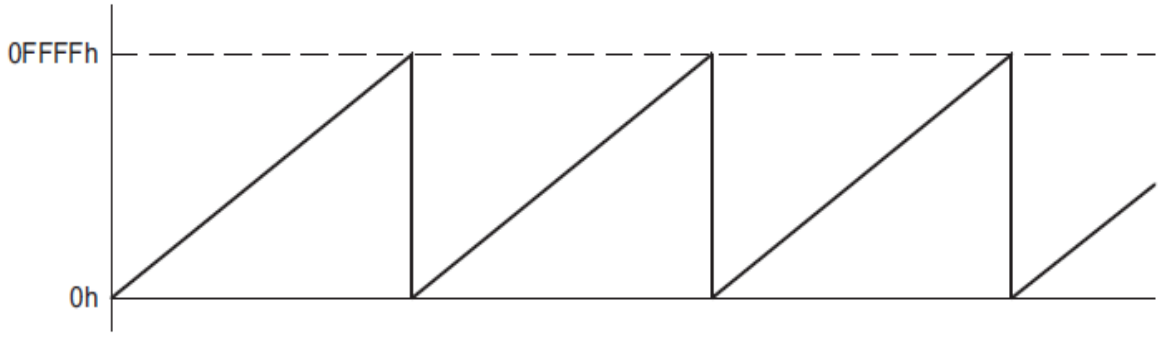
Η λειτουργία Up χρησιμοποιείται για περιόδους του Timer που διαφέρουν 0FFFFh μετρήσεις. Ο Timer επανειλημμένα μετράει έως την τιμή του καταχωρητή σύγκρισης TAxCCR0, που καθορίζει την περίοδο. Ο αριθμός μετρήσεων του Timer που μετράει στην περίοδο είναι TAxCCR0 + 1. Όταν η τιμή του Timer ισούται με τον TAxCCR0, ο Timer επαναφέρεται και μετράει πάλι από το μηδέν.



Εικόνα 44. Λειτουργία προς τα πάνω (Up Mode)

Συνεχόμενη λειτουργία (Continuous Mode):

Ο Timer μετράει επανειλημμένα από το 0 έως το 0FFFFh.

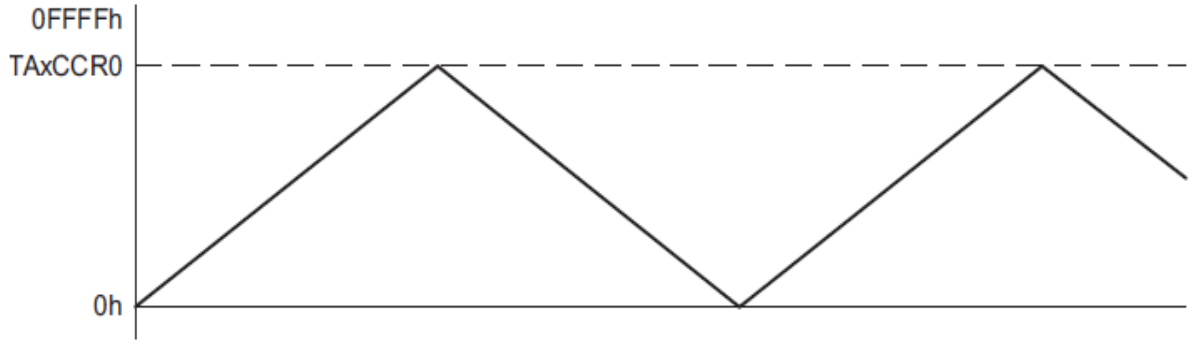


Εικόνα 45. Συνεχόμενη λειτουργία (Continuous Mode)

Λειτουργία πάνω / κάτω (Up / Down Mode):

Η λειτουργία πάνω / κάτω χρησιμοποιείται αν ή περίοδο του Timer πρέπει να είναι διαφορετική από τις μετρήσεις 0FFFFh, καθώς αν χρειάζεται συμμετρική παραγωγή παλμών.

Ο Timer μετράει επανειλημμένα από το 0 έως την τιμή του καταχωρητή σύγκρισης TAxCCR0 και προς τα πίσω μέχρι να ξαναπάει στο 0. Η περίοδος είναι διπλάσια της τιμής του καταχωρητή TAxCCR0.



Εικόνα 46. Λειτουργία πάνω / κάτω (Up / Down Mode)

Καταχωρητής ελέγχου του Timer_A [Timer_Ax Control Register (TAxCTL)].

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ID		MC		Reserved	TACLR	TAIE	TAIFG
rw-0	rw-0	rw-0	rw-0	rw-0	w-0	rw-0	rw-0

Σχήμα 16. Καταχωρητής TAxCTL

Πίνακας 23. Περιγραφή καταχωρητή TAxCTL				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 10	Reserved	RW	0h	Reserved
9 - 8	TASSEL	RW	0h	Επιλογή πηγής ρολογιού του Timer_A 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
7 - 6	ID	RW	0h	Διαιρέτης εισόδου. Αυτά τα bits μαζί με του TAIDEX τα bits επιλέγουν τον διαιρετή για το ρολόι εισόδου. 00b = /1 01b = /2 10b = /4 11b = /8
5 - 4	MC	RW	1h	Έλεγχος τρόπου λειτουργίας. MCx = 00h όταν ο Timer_A δεν χρησιμοποιείται κάνει εξοικονόμηση ενέργειας 00b = Stop mode: Timer is halted 01b = Up mode: O Timer μετράει έως την τιμή του TAxCCR0 10b = Continuous mode: O Timer μετράει έως το 0FFFFh 11b = Up/down mode: O Timer μετράει έως την τιμή του TAxCCR0 έπειτα πίσω στο 0000h
3	Reserved	RW	0h	Reserved
2	TACLR	RW	0h	Timer_A Καθαρίζεται. Ορίζοντας αυτό το bit επαναφέρει τον TAxR, του timer την λογική διαίρεσης και την διεύθυνση μέτρησης. Το TACLR bit είναι αυτόματα στο reset και διαβάζεται πάντα σαν μηδέν.
1	TAIE	RW	Oh	Ενεργοποίηση του interrupt στον Timer_A. Αυτό το bit ενεργοποιεί το αίτημα διακοπής στον TAIFG. 0b = Interrupt disabled 1b = Interrupt enabled
0	TAIFG	RW	0h	Timer_A interrupt flag 0b = No interrupt pending 1b = Interrupt pending

Καταχωρητής TAxR, Timer_Ax Counter Register

15	14	13	12	11	10	9	8
TAxR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
TAxR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 17. Καταχωρητής TAxR

Πίνακας 24. Περιγραφή καταχωρητή TAxR				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 0	TAxR	RW	0h	Καταχωρητής του Timer_A. Ο καταχωρητής TAxR εμπεριέχει τις μετρήσεις του Timer_A.

Καταχωρητές από TAxCCTL0 έως TAxCCTL6, Timer_Ax Capture/Compare Control 0 Register to Timer_Ax Capture/Compare Control 6.

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-0	rw-0	rw-0	rw-0	rw-0	r-0	r-0	rw-0
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-0	rw-0	rw-0	rw-0	r	rw-0	rw-0	rw-0

Σχήμα 18. Καταχωρητές από TAxCCTL0 έως TAxCCTL6

Πίνακας 25. Περιγραφή καταχωρητών από TAxCCTL0 έως TAxCCTL6				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture στο ανερχόμενο μέτωπο 10b = Capture στο κατερχόμενο μέτωπο 11b = Capture και στα 2 μέτωπα (ανερχόμενο και κατερχόμενο)
13 - 12	CCIS	RW	0h	Επιλογή εισόδου Capture/Compare. Αυτά τα bits επιλέγουν το σήμα εισόδου του TAxCCR0. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Συγχρονίζει την πηγή του capture. Αυτό το bit χρησιμοποιείται για να συγχρονίζει το σήμα

				εισόδου του capture με το ρολόι του timer. 0b = Ασύγχρονο capture 1b = Σύγχρονο capture
10	SCCI	RW	0h	Συγχρονισμένη είσοδος capture/compare. Το επιλεγμένο σήμα εισόδου είναι κλειδωμένο με το σήμα EQUx και μπορεί να διαβαστεί μέσω αυτού του bit.
9	Reserved	R	0h	Reserved. Διαβάζεται σαν μηδέν.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7 - 5	OUTMOD	RW	0h	Τρόπος λειτουργίας εξόδου. Τα Modes 2, 3, 6 και 7 δεν είναι χρήσιμα για τον TAxCCR0 λόγο του EQUx = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Ενεργοποίηση των interrupt του Capture/Compare. Αυτό το bit ενεργοποιεί το αίτημα του interrupt request of the για την αντίστοιχη σημαία CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Είσοδος του Capture/Compare. Το επιλεγμένο σήμα εισόδου μπορεί να διαβαστεί από αυτό το bit.
2	OUT	RW	0h	Έξοδος. Για τον τρόπο λειτουργία εξόδου 0, αυτό το bit απευθείας ελέγχει την κατάσταση της εξόδου. 0b = Output low 1b = Output high
1	COV	RW	0h	Υπερχείλιση (overflow) Capture. Αυτό το bit υποδεικνύει ότι υπήρξε (occurred) capture overflow. Το COV πρέπει να γίνει reset με το λογισμικό. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/Compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

Καταχωρητές από TAxCCR0 έως TAxCCR6, Timer_Ax Capture/Compare 0 Register to Timer_Ax Capture/Compare 6 Register.

15	14	13	12	11	10	9	8
TAxCCRn							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
TAxCCRn							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Σχήμα 19. Καταχωρητές από TAxCCR0 έως TAxCCR6

Πίνακας 26. Περιγραφή καταχωρητών από TAxCCR0 έως TAxCCR6				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 0	TAxCCRn (TAxCCR0 έως TAxCCR6)	RW	0h	**Compare mode:** TAxCCRn κρατάει τα δεδομένα για σύγκριση στην τιμή του timer του καταχωρητή Timer_A, TAxR. **Capture mode:** Ο καταχωρητής Timer_A, TAxR, αντιγράφεται μέσα στον καταχωρητή TAxCCRn όταν το capture πραγματοποιείται.

Καταχωρητής TAxIV, Timer_Ax Interrupt Vector Register

15	14	13	12	11	10	9	8
TAIV							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
TAIV							
r0	r0	r0	r0	r-0	r-0	r-0	r0

Σχήμα 20. Καταχωρητής TAxIV

Πίνακας 27. Περιγραφή του καταχωρητή TAxIV				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 0	TAIV	R	0h	Timer_A interrupt vector value **00h** = No interrupt

				pending **02h** = Interrupt Source: Capture/compare 1; Interrupt Flag: TAxCCR1 CCIFG; Interrupt Priority: Highest **04h** = Interrupt Source: Capture/compare 2; Interrupt Flag: TAxCCR2 CCIFG **06h** = Interrupt Source: Capture/compare 3; Interrupt Flag: TAxCCR3 CCIFG **08h** = Interrupt Source: Capture/compare 4; Interrupt Flag: TAxCCR4 CCIFG **0Ah** = Interrupt Source: Capture/compare 5; Interrupt Flag: TAxCCR5 CCIFG **0Ch** = Interrupt Source: Capture/compare 6; Interrupt Flag: TAxCCR6 CCIFG **0Eh** = Interrupt Source: Timer overflow; Interrupt Flag: TAxCTL TAIFG; Interrupt Priority: Lowest

Καταχωρητής TAxEX0, Timer_Ax Expansion 0 Register.

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved					TAIDEX(1)		
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0

Σχήμα 21. Καταχωρητής TAxEX0

Πίνακας 28. Περιγραφή του καταχωρητή TAxEX0				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 3	Reserved	R	0h	Reserved. Διαβάζετε σαν μηδέν.
2 - 0	TAIDEX	RW	0h	Διαιρέτης επέκτασης

				εισόδου (Input divider expansion). Αυτά τα bits μαζί με τα ID bits επιλέγουν τον διαιρέτη για ρολόι εισόδου. 000b = Διαίρεση δια 1 (Divide by 1) 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8

Ένα παράδειγμα του Timer_A, που αναβοσβήνει μέσω interrupt το κόκκινο LED του αναπτυξιακού MSP-EXP432P401R ανά 1 δευτερόλεπτο.

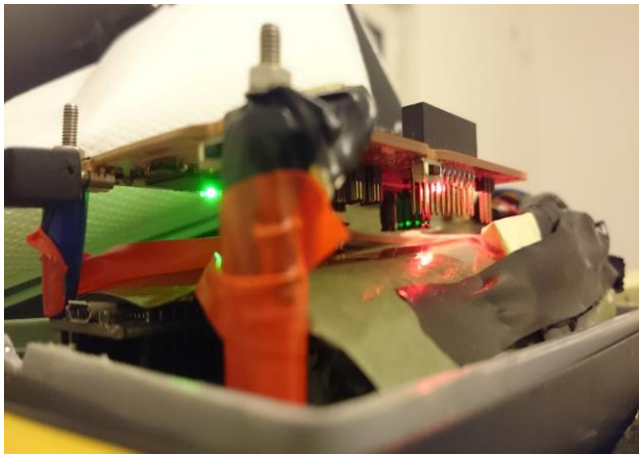
$$t_{red} = \frac{1}{\left(\frac{f_{clock}}{prescalers\ ID * IDEX}\right)} \times TA1CCR0 = \frac{1}{\left(\frac{3 \times 10^6 Hz}{8 * 8}\right)} \times (46875) = 1\ Second$$

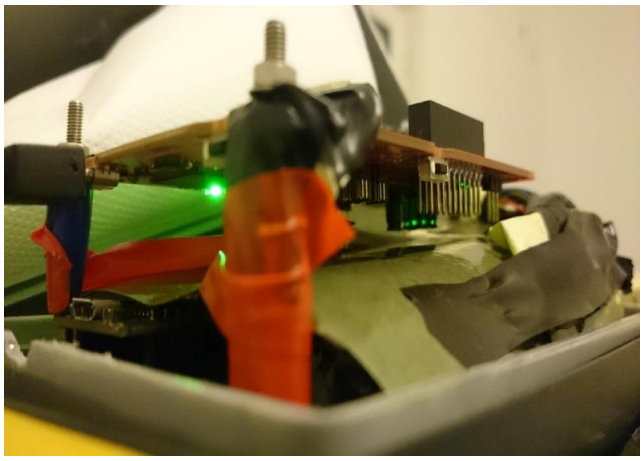
Κώδικας CCS σε γλώσσα προγραμματισμού C:

```
#include "msp.h"

int main(void)
{
  __disable_irq();

  //Arxikopoihsh tou kokkinou LED P2.0
  P2->SEL1 &= ~1; //Ru8misei tou P2.0 san aplh I/O
  P2->SEL0 &= ~1;
  P2->DIR |= 1; //P2.0 orismos san e3odos

  TIMER_A1->CTL = 0x02D1; //SMCLK, ID=/8, up mode, TA clear
  TIMER_A1->CCR[0] = 46875 - 1; //ru8misei gia 1 sec
  TIMER_A1->EX0 = 7; //deuteros prescaler IDEX = /8
  TIMER_A1->CCTL[0] |= 0x10; // energopoihsh tou interrupt TA1_0

  NVIC_SetPriority(TA1_0_IRQn, 3); //8eto thn proteraiothta sto 3 ston NVIC
  NVIC_EnableIRQ(TA1_0_IRQn); //energopoihsh interrupt ston NVIC

  __enable_irq(); //genikh energopoihsh ton IRQs

  while (1) //atermon broxos
  {
  }
}

void TA1_0_IRQHandler(void) //ana 1 sec
{
```

```
  TIMER_A1->CCTL[0] &= ~1; //ka8arismos interrupt flag
  P2->OUT ^= 1; //anabosbhnei to red LED
}
```

Φωτογραφίες από την λειτουργία του προγράμματος





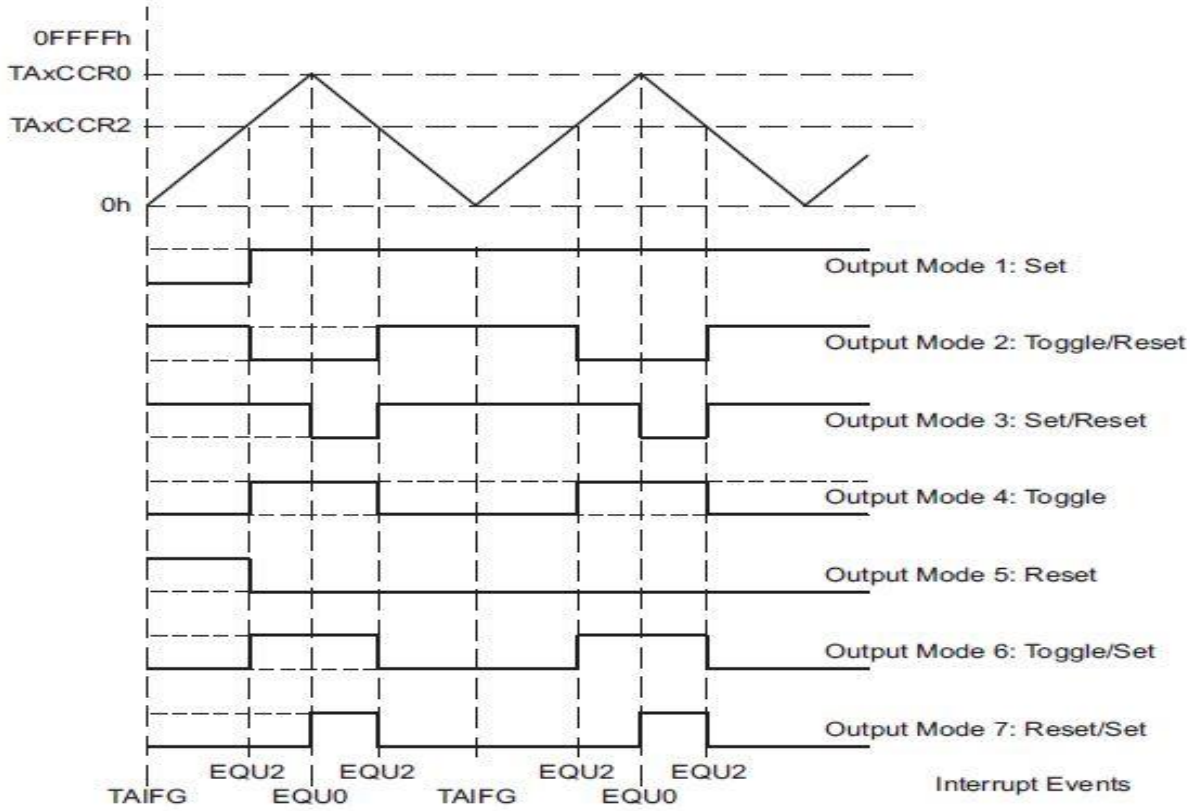
Παραγωγή κυματομορφής PWM

Με τον τρόπο λειτουργίας σύγκρισης (Compare Mode) του Timer_A μας δίνεται η δυνατότητα να παράγουμε κυματομορφή Pulse Width Modulation (PWM) στην έξοδο (ή και interrupts σε συγκεκριμένα χρονικά διαστήματα, όπως είδαμε και στα προηγούμενα παραδείγματα). Το compare mode επιλέγεται από το bit CAP = 0 στον καταχωρητή TAxCCTL. Όταν ο TAxR μετρήσει την τιμή που έχει ο TAxCCRn (το n είναι οι καταχωρητές από 0 έως 6), τότε γίνονται τα εξής:

- Η σημαία διακοπής CCIFG ορίζεται
- Το εσωτερικό σήμα EQUn =1
- Το σήμα EQUn επηρεάζει την έξοδο ανάλογα με τον τρόπο λειτουργίας της εξόδου (output mode)
- Το σήμα εισόδου CCI είναι μανδαλώνεται σε SCCI.

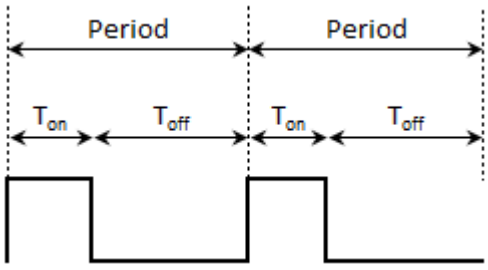
Παράδειγμα εξόδου του Timer σε Up/Down λειτουργία

Τα σήματα εξόδου αλλάζουν όταν ο Timer γίνεται ίσως με τον TAxCCRn σε οποιαδήποτε κατεύθυνση μέτρησης (Up ή Down) και όταν ο Timer ισούται με TAxCCR0, εξαρτάται από τον τρόπο λειτουργίας της εξόδου (output mode).



Εικόνα 47. Παράδειγμα εξόδου σε Timer Up/Down Mode χρησιμοποιώντας τους καταχωρητές TAxCCR0 και TAxCCR2

Άρα παρατηρείτε ότι κάποιες λειτουργίες εξόδου μπορούν να δημιουργήσουν την PWM κυματομορφή.



Εικόνα 48. Κυματομορφή PWM

Η PWM κυματομορφή έχει συχνότητα, περίοδο και Duty Cycle που στην Up/Down (δηλαδή επί 2 η τιμή του TAxCCR0 για τον υπολογισμό συχνότητας και περιόδου) λειτουργία και Toggle/Reset Output Mode 2, υπολογίζονται ως εξής:

$$F_{PWM} = \frac{F_{clock}}{(TAxCCR0 \times 2)}$$

$$T_{PWM} = \frac{1}{F_{PWM}} \text{ ή } T_{PWM} = \frac{(TAxCCR0 \times 2)}{F_{clock}}$$

$$Duty\ Cycle = \frac{TAxCCR2}{TAxCCR0} \text{ ή } Duty\ Cycle = \frac{T_{on}}{(T_{on} + T_{Off})} = \frac{T_{on}}{T_{PWM}}$$

$$Duty\ Cycle(\%) = \frac{TAxCCR2}{TAxCCR0} \times 100$$

Παρατηρείτε ότι η περίοδος του PWM εξαρτάται από την τιμή του TAxCCR0 (μέγιστη τιμή 2^16-1 = 65535) και το Duty Cycle από την τιμή του δεύτερου καταχωρητή που κάνει την σύγκριση παράδειγμα TAxCCR2 που φαίνεται και στην Εικόνα 47.

Ένα παράδειγμα του Timer_A να δημιουργεί PWM κυματομορφή με συχνότητα 30 Hz και με duty cycle 33.33% και να την δίνει στο pin P2.7 ή αλλιώς νούμερο pin 40 και να λειτουργεί ένα μπλε LED, χρησιμοποιήθηκε η λειτουργία εξόδου 2 (Output Mode 2).

Υπολογισμοί συχνότητας και duty cycle:

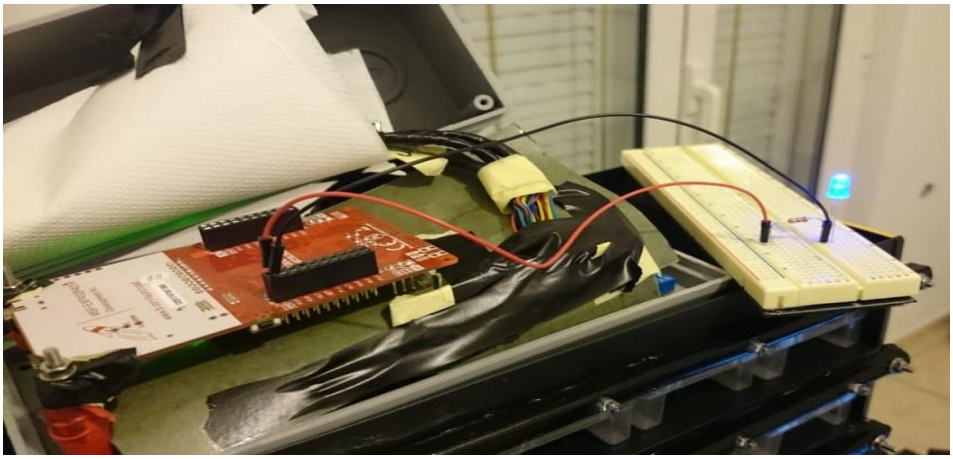
$$F_{PWM} = \frac{F_{clock}}{(TAxCCR0 \times 2)} = \frac{3 \times 10^6\ Hz}{(50000 \times 2)} = 30\ Hz$$

$$Duty\ Cycle(\%) = \frac{TAxCCR4}{TAxCCR0} \times 100 = \frac{16667}{50000} \times 100 = 33.33\%$$

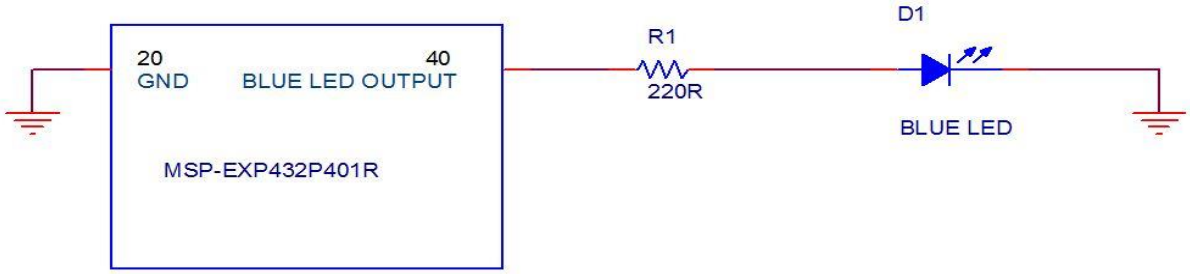
Συνδυαστικός κώδικας CCS και Energia σε γλώσσα προγραμματισμού C, το πρόγραμμα γράφτηκε στον Energia IDE και έγινε import στον CCS IDE:

```
#include "msp.h" //Eisagw tin Biblio8hkh tou Msp

void setup() {
  Serial.begin(115200);
  //Kanw tin 8yra P2.7 Timer A0.4 kai to dilono san e3odo
  P2SEL0 |= 0x80; //0b 1000 0000 to 7 pinaki to maskaro me OR san asso ta alla den
peirazontai
  P2->SEL1 &= ~0x80; //0b 0111 1111 to 7 pinaki to maskaro me AND san 0 ta alla
den peirazontai etsi exw epila3i sto SEL0 = 1 kai sto SEL1 = 0; ara 01 epilogi ara
primary apo to user guide
  P2->DIR |= 0x80; // 0b 1000 0000 san OUTPUT to pin 7 ta alla apeirakta

  //Ru8mizw ton TimerA0.4 san PWM syxnothta: 3 000 000 HZ / 50000 / 2 = 30 Hz
  TIMER_A0->CCR[0] = 50000 - 1;        //PWM Periodo
  TIMER_A0->CCR[4] = (50000 - 1) / 3; //CCR4 PWM duty cycle
  TIMER_A0->CCTL[4] = 0x40;            // To CCR4 toggle/reset mode
  TIMER_A0->CTL = 0x0234;              //Xrhsimopoiw SMCLK (3 MHZ Standby ru8misi),
up/down mode, clear TA0R register
  }

void loop() {
  Serial.println(TA0R); // Diabazw ton TA0 Counter
}
```

Φωτογραφία λειτουργίας του προγράμματος



Χρησιμοποιήθηκε το Σχηματικό 1.



Σειριακή διεπαφή περιφερειακών [Serial Peripheral Interface (SPI)]

Το SPI ανήκει στις Ενισχυμένες διεπαφές διεθνούς σειριακής επικοινωνίας [Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B)].

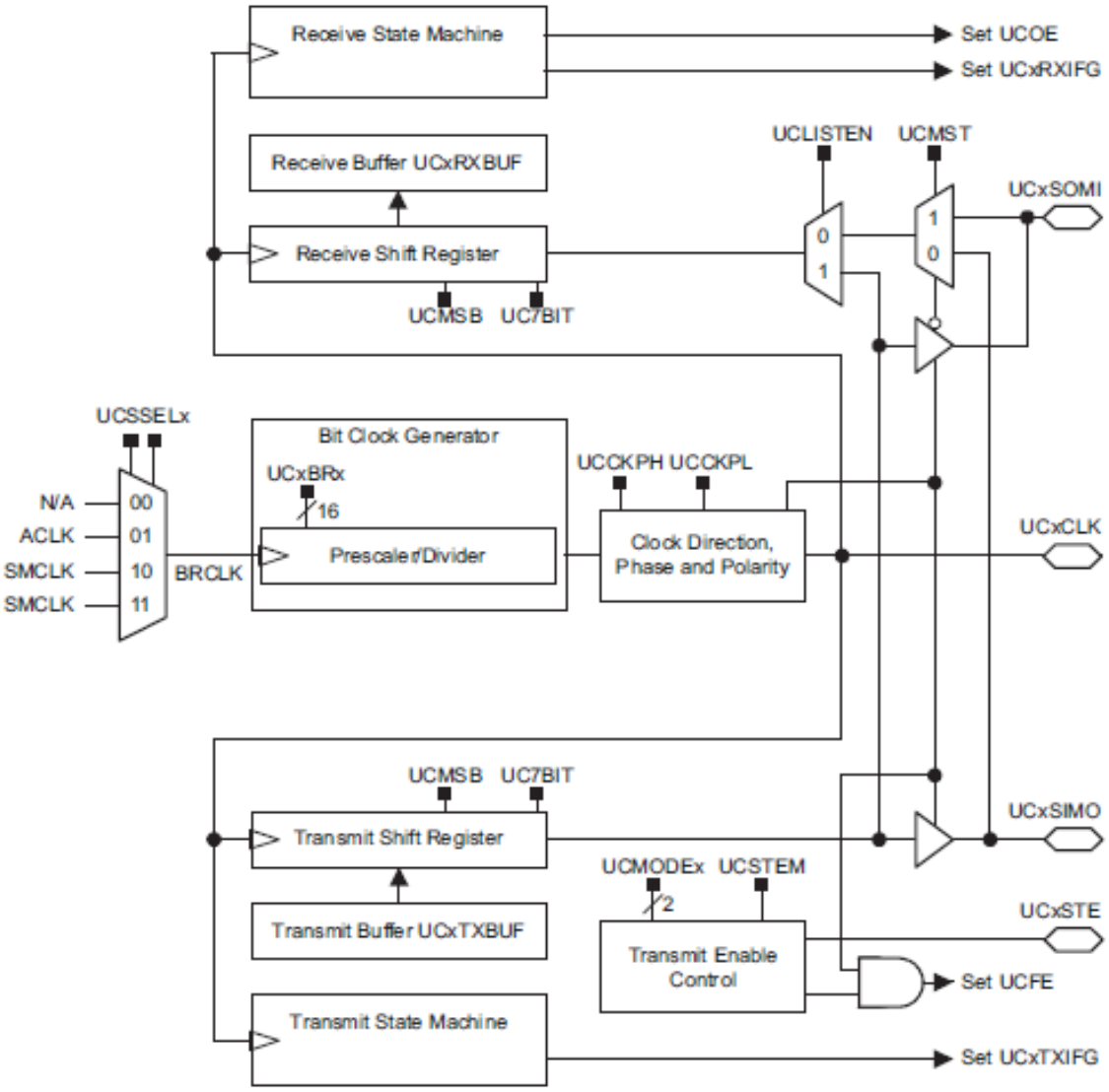
Η λειτουργία του SPI mode:

Στον σύγχρονο (synchronous) τρόπο λειτουργίας (mode), το eUSCI συνδέει τη συσκευή με ένα εξωτερικό σύστημα, μέσω τριών ή τεσσάρων pins: UCxSIMO, UCxSOMI, UCxCLK και UCxSTE. Ο τρόπος λειτουργίας SPI επιλέγεται όταν ορίζεται το UCSYNC bit και συγκεκριμένα το SPI mode 3 pin ή 4 pin επιλέγεται με τα bits UCMODEx.

Χαρακτηριστικά που περιλαμβάνονται στο SPI mode:

- 7 bit ή 8 bit μήκος δεδομένων (data length)
- Λιγότερο σημαντικό ψηφίο πρώτο [Less Significant Bit (LSB) first] ή σημαντικότερο ψηφίο πρώτο [Most Significant Bit (MSB) first] μετάδοση (transmit) και λήψη (receive) δεδομένων
- 3 pin και 4 pin λειτουργίας SPI

- Τρόποι λειτουργίας αφέντη (Master) ή σκλάβου (Slave)
- Ανεξάρτητοι καταχωρητές ολίσθησης (μετατόπισης) μετάδοσης (transmit shift registers) και ολίσθησης (μετατόπισης) λήψης (receive shift registers)
- Ξεχωριστοί καταχωρητές buffer (προσωρινής μνήμης) για μετάδοση και λήψη
- Συνεχόμενη λειτουργία μετάδοσης και λήψης
- Επιλογή πολικότητας ρολογιού (clock polarity) και έλεγχος φάσης (phase control)
- Προγραμματιζόμενη συχνότητα ρολογιού στο master mode
- Ανεξάρτητη δυνατότητα διακοπής για λήψη και μετάδοση



Εικόνα 49. Μπλοκ διάγραμμα eUSCI σε τρόπο λειτουργίας SPI

Στο SPI mode, τα δεδομένα μεταδίδονται ή λαμβάνονται σειριακά από πολλαπλές συσκευές χρησιμοποιώντας ένα κοινό ρολόι που παρέχεται από τον αφέντη (Master), ένα επιπρόσθετο pin ελέγχεται από τον Master είναι το UCxSTE και παρέχεται για να ενεργοποιεί την συσκευή ή τις συσκευές (Slaves) για να λαμβάνουν και να μεταδίδουν δεδομένα.

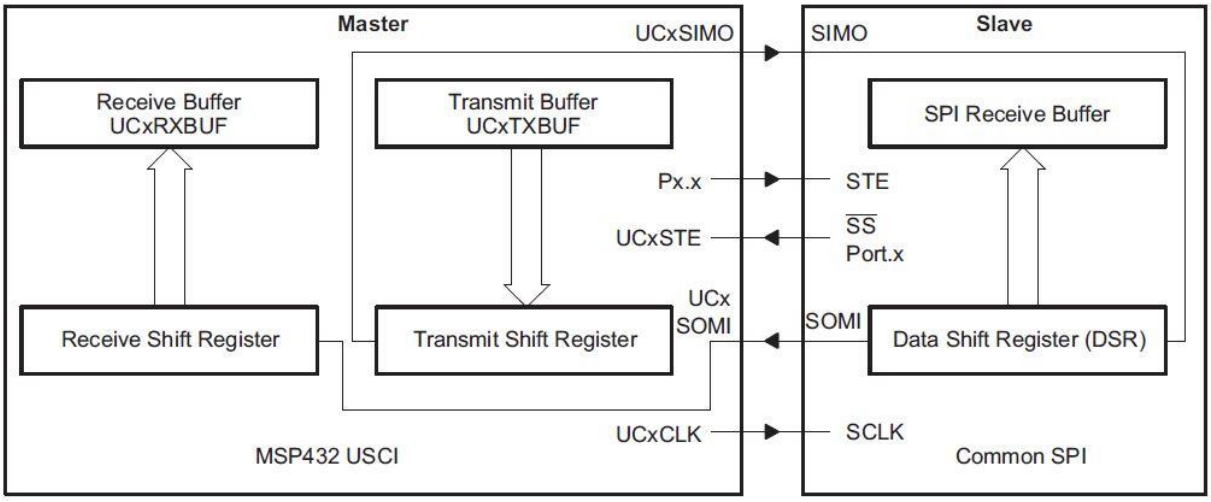
Τρία ή τέσσερα σήματα χρησιμοποιούνται στο SPI για την ανταλλαγή δεδομένων:

- UCxSIMO: σκλάβος είσοδος (slave in), αφέντης έξοδος (master out)
    - Τρόπος λειτουργίας αφέντη (Master mode): UCxSIMO είναι η γραμμή εξόδου δεδομένων.
    - Τρόπος λειτουργίας σκλάβου (Slave mode): UCxSIMO είναι η γραμμή εισόδου δεδομένων.
- UCxSOMI: σκλάβος έξοδος (slave out), αφέντης είσοδος (master in)
    - Master mode: UCxSOMI είναι η γραμμή εισόδου δεδομένων.
    - Slave mode: UCxSOMI είναι η γραμμή εξόδου δεδομένων.
- UCxCLK: eUSCI SPI clock
    - Master mode: UCxCLK είναι μια έξοδος
    - Slave mode: UCxCLK είναι μια είσοδος
- UCxSTE: slave transmit enable
    - Χρησιμοποιείται στο 4-pin mode και επιτρέπει πολλαπλούς αφέντες σε έναν μονό δίαυλο. Δεν χρησιμοποιείται στο 3-pin mode

Τρόπος λειτουργία αφέντη (Master mode):

Η Εικόνα 50. δείχνει το eUSCI σαν αφέντη και τις δύο ρυθμίσεις των 3-pin και των 4-pin. Το eUSCI ενεργοποιεί τη μεταφορά δεδομένων όταν τα δεδομένα μετακινούνται στην προσωρινή μνήμη δεδομένων (data buffer) εκπομπής (transmit) UCxTXBUF. Τα δεδομένα του UCxTXBUF μετακινούνται στον καταχωρητή ολίσθησης [μετατόπισης (shift)] μετάδοσης (TX), όταν ο καταχωρητής TX ολίσθησης(μετατόπισης) μετάδοσης είναι άδειος, ενεργοποιεί την μεταφορά δεδομένων στο UCxSIMO ξεκινώντας είτε με το MSB είτε με το LSB, εξαρτάται από της ρυθμίσεις του UCMSB. Τα δεδομένα στο UCxSOMI μετατοπίζονται στον καταχωρητή μετατόπισης λήψης (receive shift register) στην αντίθετη ακμή ρολογιού. Όταν λαμβάνεται ο χαρακτήρας (character), τα δεδομένα λήψης μετακινούνται από τον καταχωρητή μετατόπισης λήψης (RX) στην αποθηκευμένη προσωρινή μνήμη δεδομένων

UCxRXBUF και λαμβάνει ένα interrupt flag και ορίζεται το UCRXIFG υποδεικνύοντας ότι η λειτουργία RX ή TX έχει ολοκληρωθεί.



Εικόνα 50. Σε λειτουργία Master το eUSCI και εξωτερικός Slave (UCSTEM = 0)

Για να λάβεις δεδομένα στο eUSCI Master mode, τα δεδομένα πρέπει να γραφτούν στο UCxTXBUF, επειδή οι λειτουργίες μετάδοσης και λήψης λειτουργούν ταυτόχρονα.

Υπάρχουν δύο διαφορετικές ρυθμίσεις στο eUSCI σαν a 4-pin master που είναι:

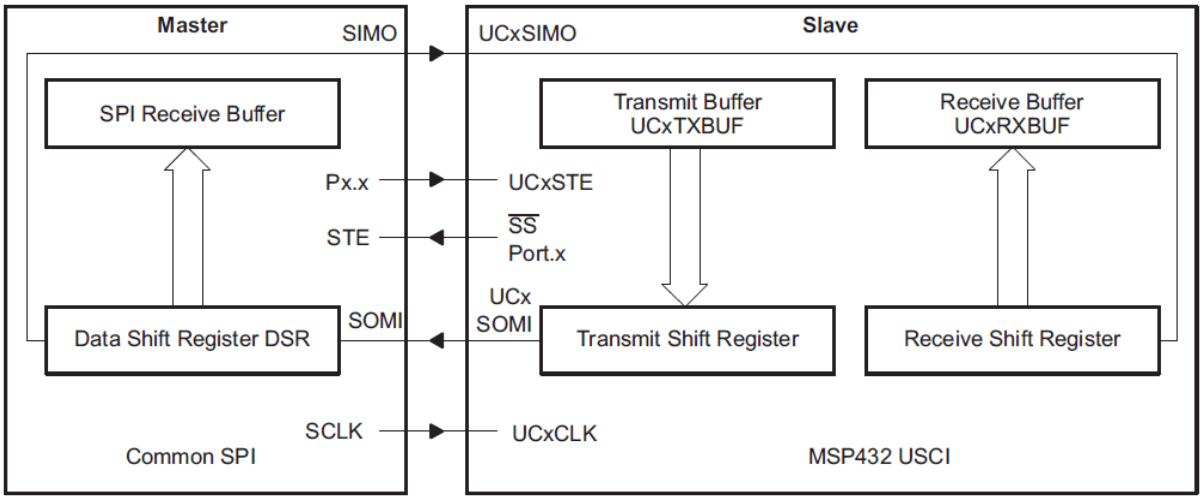
- Το τέταρτο pin χρησιμοποιείται σαν είσοδος για να αποφύγει συγκρούσεις με άλλους Masters (UCSTEM = 0).
- Το τέταρτο pin χρησιμοποιείται σαν έξοδος για να παράγει ένα σήμα ενεργοποίησης για τον Slave (UCSTEM = 1).

Το bit UCSTEM χρησιμοποιείτε για την επιλογή του αντίστοιχου τρόπου λειτουργίας.

Τρόπος λειτουργία σκλάβου (Slave mode):

Η Εικόνα 51. δείχνει το eUSCI σαν σκλάβο και τις δύο ρυθμίσεις των 3-pin και των 4-pin. Το UCxCLK χρησιμοποιείται σαν είσοδος για το ρολόι του SPI και πρέπει να παρέχεται από εξωτερικό αφέντη (Master). Ο ρυθμός μεταφοράς των δεδομένων καθορίζεται από αυτό το ρολόι και όχι από την εσωτερική γεννήτρια bit ρολογιού (internal bit clock generator). Τα δεδομένα γράφονται στην προσωρινή μνήμη (buffer) UCxTXBUF και μεταφέρονται στον καταχωρητή TX shift register πριν ξεκινήσει το UCxCLK μεταδίδονται στο UCxSOMI. Τα δεδομένα στο UCxSIMO μετατοπίζονται στον receive shift register στην αντίθετη ακμή του UCxCLK και μετακινούνται στην προσωρινή μνήμη UCxRXBUF όταν λαμβάνεται ο καθορισμένος αριθμός από bits. Όταν τα δεδομένα μετακινούνται από τον RX shift register στην προσωρινή μνήμη UCxRXBUF, Η σημαία διακοπής UCRXIFG ορίζεται

υποδεικνύοντας ότι τα δεδομένα έχουν ληφθεί. Το UCOE bit σφάλματος υπέρβασης ορίζεται όταν τα προηγούμενα ληφθέντα δεδομένα δεν διαβάζονται από το UCxRXBUF προτού μεταφερθούν νέα δεδομένα στο UCxRXBUF.



Εικόνα 51. Σε λειτουργία Slave το eUSCI και εξωτερικός Master

Έλεγχος του σειριακού ρολογιού (Serial Clock Control).

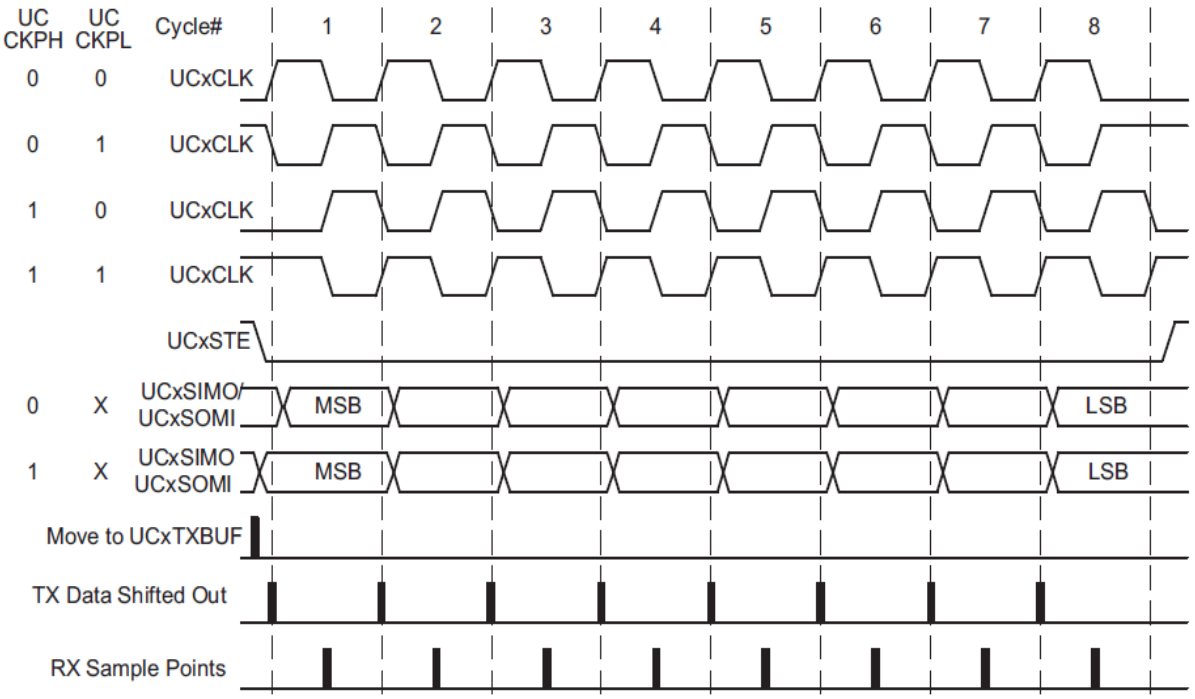
Το σειριακό UCxCLK παρέχεται από τον master στον SPI δίαυλο. Όταν το UCMST = 1, το ρολόι των bit παρέχεται από την eUSCI γεννήτρια bit ρολογιού στο UCxCLK pin. Το ρολόι που χρησιμοποιείται για να φτιάξει παλμούς ρολογιού επιλέγεται από τα UCSSELx bits. Όταν το UCMST = 0, το eUSCI ρολόι παρέχεται στο UCxCLK pin από τον master, η γεννήτρια παραγωγής παλμών ρολογιού δεν χρησιμοποιείτε και τα the UCSSELx bits δεν μας ενδιαφέρουν. Ο SPI δέκτης και πομπός λειτουργούν παράλληλα και χρησιμοποιούν την ίδια πηγή ρολογιού για την μεταφορά δεδομένων.

Η τιμή των 16-bits στους καταχωρητές έλεγχου ρυθμού bit UCBRx (UCxxBR1 και UCxxBR0) είναι ο συντελεστής διαίρεσης της πηγής ρολογιού eUSCI, BRCLK.

Το μέγιστο ρολόι bit που μπορεί να δημιουργηθεί στη λειτουργία master είναι το BRCLK. Η διαμόρφωση δεν χρησιμοποιείται στον τρόπο λειτουργίας SPI και το UCAxMCTLW πρέπει να καθαριστεί όταν χρησιμοποιείται ο τρόπος λειτουργίας SPI για το eUSCI_A. Η συχνότητα ρολογιού του UCAxCLK ή UCBxCLK δίνεται από το μαθηματικό τύπο: $f_{BitClock} = \frac{f_{BRCLK}}{UCBRx}$

Πολικότητα και φάση σειριακού ρολογιού (Serial Clock Polarity and Phase).

Η πολικότητα και η φάση του UCxCLK ρυθμίζονται ανεξάρτητα με τα bit ελέγχου UCCKPL και UCCKPH του eUSCI. Η Εικόνα 52. μας δείχνει το χρονοδιάγραμμα για κάθε περίπτωση.



Εικόνα 52. Διάγραμμα χρονισμού του eUSCI SPI, με UCMSB = 1

Καταχωρητής ελέγχου του eUSCI_Bx, UCBxCTLW0 Register, eUSCI_Bx Control Register 0

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
7	6	5	4	3	2	1	0
UCSSELx		Reserved				UCSTEM	UCSWRST
rw-1	rw-1	r0	rw-0	rw-0	rw-0	rw-0	rw-1
	Modify only when UCSWRST = 1.						

Σχήμα 22. Καταχωρητής ελέγχου του SPI, UCBxCTLW0

Πίνακας 29. Περιγραφή του καταχωρητή UCBxCTLW0				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15	UCCKPH	RW	0h	Επιλογή φάσης ρολογιού (Clock phase select) 0b = Τα δεδομένα αλλάζουν στην πρώτη άκρη του UCLK και καταγράφονται στην επόμενη άκρη. 1b = Τα δεδομένα καταγράφονται στην πρώτη άκρη του UCLK και αλλάζουν στην επόμενη άκρη.
14	UCCKPL	RW	0h	Επιλογή πολικότητας ρολογιού (Clock polarity select)

				0b= Η ανενεργή κατάσταση είναι η low. 1b= Η ανενεργή κατάσταση είναι η high.
13	UCMSB	RW	0h	Επιλογή πρώτου σημαντικότερου ψηφιού (MSB first select). Ελέγχει την κατεύθυνση του καταχωρητή μετατόπισης λήψης και μετάδοσης. 0b= LSB first 1b= MSB first
12	UC7BIT	RW	0h	Μήκος χαρακτήρα (Character length). Επιλέγει το μήκος των χαρακτήρων σε 7 bit ή 8 bit. 0b= 8-bit data 1b= 7-bit data
11	UCMST	RW	0h	Επιλογή τρόπου λειτουργίας αφέντη (Master mode select) 0b= Slave mode 1b= Master mode
10 - 9	UCMODEx	RW	0h	eUSCI mode. Τα bit του UCMODEx επιλέγουν τον σύγχρονο τρόπο λειτουργίας όταν UCSYNC = 1. **00b** = 3-pin SPI **01b** = 4-pin SPI με UCxSTE ενεργό στο high: Ενεργοποίηση του Slave όταν UCxSTE = 1 **10b** = 4-pin SPI με UCxSTE ενεργό στο low:Ενεργοποίηση του Slave όταν UCxSTE = 0 **11b** = I2C mode
8	UCSYNC	RW	0h	Ενεργοποίηση του σύγχρονου τρόπου λειτουργίας (Synchronous mode enable) 0b= Ασύγχρονο mode 1b= Σύγχρονο mode
7 - 6	UCSSELx	RW	0h	Επιλογή πηγής ρολογιού eUSCI (eUSCI clock source select). Αυτά τα bits επιλέγουν την πηγή ρολογιού BRCLK στο master mode. Το UCxCLK πάντα χρησιμοποιείται στο slave mode. 00b= Reserved 01b= ACLK 10b= SMCLK 11b= SMCLK
5 - 2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	Το STE mode επιλέγεται στο master mode (STE mode select in master mode). Αυτό το bit αγνοείται στον slave ή στον τρόπο λειτουργίας 3 καλωδίων (3-wire mode). 0b= Το STE pin χρησιμοποιείται για την αποφυγή συγκρούσεων με άλλους masters 1b= Το STE pin

				χρησιμοποιείται για να παράγει σήμα ενεργοποίησης για τον 4 - wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Απενεργοποίηση. Το reset στο eUSCI απελευθερώθηκε για να λειτουργήσει. 1b = Ενεργοποίηση. Η λογική eUSCI διατηρείται σε κατάσταση επαναφοράς.

Καταχωρητής έλεγχου ρυθμού των bit UCBxBRW, eUSCI_Bx Bit Rate Control Register 1

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
Modify only when UCSWRST = 1.							

Σχήμα 23. Καταχωρητής έλεγχου ρυθμού των bit UCBxBRW

Πίνακας 30. Περιγραφή του καταχωρητή UCBxBRW				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 0	UCBRx	RW	0h	Ρυθμίσεις διαιρέτη ρολογιού παλμών (Bit clock prescaler setting)

Καταχωρητής κατάστασης UCBxSTATW του eUSCI_Bx, eUSCI_Bx Status Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	Reserved				UCBUSY
rw-0	rw-0	rw-0	r0	r0	r0	r0	r-0
Modify only when UCSWRST = 1.							

Σχήμα 24. Καταχωρητής κατάστασης UCBxSTATW

Πίνακας 31. Περιγραφή του καταχωρητή UCBxSTATW				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. Το UCLISTEN bit επιλέγει

				το loopback mode. 0b = Disabled 1b = Enabled Η έξοδος του πομπού ανατροφοδοτείτε εσωτερικά πίσω προς τον δέκτη.
6	UCFE	RW	0h	Σημαία σφάλματος πλαισίου (Framing error flag). Αυτό το bit υποδεικνύει μια σύγκρουση στον δίαυλο 4-wire master mode. Το UCFE δεν χρησιμοποιείται στο 3-wire master ή σε οποιοδήποτε slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Σημαία σφάλματος υπέρβασης (Overrun error flag). Αυτό το bit ορίζεται (set) όταν ένας χαρακτήρας μεταφέρεται στην προσωρινή μνήμη λήψης UCxRXBUF πριν διαβαστεί ο προηγούμενος χαρακτήρας. Το UCOE καθαρίζεται αυτόματα όταν το UCxRXBUF διαβάζεται και δεν πρέπει να καθαρίζεται (clear) από το λογισμικό, αλλιώς δεν λειτουργεί σωστά. 0b = No error 1b = Overrun error occurred
4 - 1	Reserved	R	0h	Reserved
0	UCBUSY	R	0h	Το eUSCI είναι απασχολημένο (eUSCI busy). Αυτό το bit υποδεικνύει εάν βρίσκεται σε εξέλιξη μια λειτουργία εκπομπής ή λήψης. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

Καταχωρητής προσωρινής μνήμης λήψης UCBxRXBUF, eUSCI_Bx Receive Buffer Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Σχήμα 25. Καταχωρητής προσωρινής μνήμης λήψης UCBxRXBUF

Πίνακας 32. Περιγραφή του καταχωρητή UCBxRXBUF				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 8	Reserved	R	0h	Reserved
7 - 0	UCRXBUFx	R	0h	Ο καταχωρητής προσωρινής μνήμης λήψης είναι προσβάσιμος από τον χρήστη και περιέχει τον τελευταίο ληφθέντα χαρακτήρα από τον καταχωρητή μετατόπισης λήψης. Διαβάζοντας τον UCxRXBUF επαναφέρει τα receive error bits και την UCRXIFG. Στο 7-bit data mode, το UCxRXBUF είναι δικαιολογημένα LSB και το MSB είναι πάντα reset.

Καταχωρητής προσωρινής μνήμης μετάδοσης UCBxTXBUF, eUSCI_Bx Transmit Buffer Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

Σχήμα 26. Καταχωρητής προσωρινής μνήμης μετάδοσης UCBxTXBUF

Πίνακας 33. Περιγραφή του καταχωρητή UCBxTXBUF				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 8	Reserved	R	0h	Reserved
7 - 0	UCTXBUFx	RW	0h	Ο καταχωρητής προσωρινής μνήμης μετάδοσης είναι προσβάσιμος από τον χρήστη και κρατάει τα δεδομένα που πρόκειται να μετακινηθούν στον καταχωρητή

				μετατόπισης μετάδοσης για να μεταδοθούν. Γράφοντας στην προσωρινή μνήμη μετάδοσης καθαρίζει την UCTXIFG. Το MSB του UCxTXBUF δεν χρησιμοποιείται στο 7-bit data και είναι reset.

Καταχωρητής ενεργοποίησης διακοπών UCBxIE, eUSCI_Bx Interrupt Enable Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

Σχήμα 27. Καταχωρητής ενεργοποίησης διακοπών UCBxIE

Πίνακας 34. Περιγραφή του καταχωρητή UCBxIE				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled.
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

Καταχωρητής σημαίας διακοπής UCBxIFG, eUSCI_Bx Interrupt Flag Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

Σχήμα 28. Καταχωρητής σημαίας διακοπής UCBxIFG

Πίνακας 35. Περιγραφή του καταχωρητή UCBxIFG				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. Η UCTXIFG ορίζεται όταν το UCxxTXBUF είναι άδειο 0b = No interrupt pending 1b = Interrupt pending

0	UCRXIFG	RW	0h	Receive interrupt flag. H UCRXIFG ορίζεται όταν το UCxxRXBUF έχει λάβει έναν πλήρη χαρακτήρα. 0b = No interrupt pending 1b = Interrupt pending

Καταχωρητής UCBxIV, eUSCI_Bx Interrupt Vector Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

Σχήμα 29. Καταχωρητής UCBxIV

Πίνακας 36. Περιγραφή του καταχωρητή UCBxIV				
Bit	Πεδίο (Field)	Τύπος (Type)	Reset	Περιγραφή
15 - 0	UCIVx	R	0h	Τιμή του eUSCI interrupt vector **0000h** = No interrupt pending **0002h** = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: **Highest** **0004h** = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: **Lowest**

Για παράδειγμά χρήσης του SPI βλέπε στον κώδικα Slave_msp432_Rafiera της Ραφιέρας στο παράρτημα Α. Επίσης δεν έγινε αναφορά καταχωρητών των αναλογικών θυρών γιατί χρησιμοποιούνται ακριβώς το ίδιο αλλά με μεγαλύτερη ευκολία στο Energia IDE.

6ο Κεφάλαιο. Οδηγός χρήσης του Energia IDE και ανάλυση βασικών εντολών του

Οδηγός χρήσης του Energia IDE v1.6.10E18

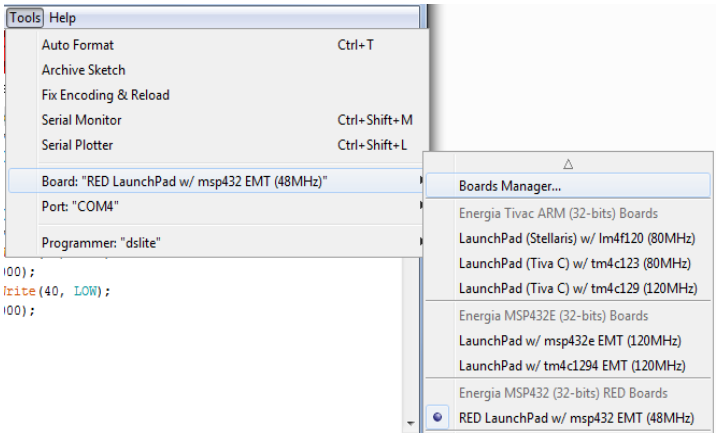
Σε αυτό το κεφάλαιο θα αναλυθούν τα βασικά εργαλεία του Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης (IDE) Energia IDE που ξεκίνησε από τον Robert Wessels τον Ιανουάριο του 2012, αυτό που κάνει το Energia IDE είναι να συνδέει την δομή (framework) του Arduino με τους μικροελεγκτές της Texas Instruments. Το Energia είναι μια πλατφόρμα open source, είναι φορητή (portable), επίσης μπορεί να λειτουργήσει σε διάφορα λειτουργικά συστήματα (Cross Platform) όπως Mac OS, Windows και Linux. Το Energia χρησιμοποιείται για την απλότητα του, ως αποτέλεσμα την μεγάλη ταχύτητα υλοποίησης εφαρμογών και μπορεί με την δυνατότητα του εισαγωγής βιβλιοθηκών να συνδυαστεί με το CCS IDE, έτσι συνδυάζεται η απλότητα (συναρτήσεις / εντολές έτοιμες) με την λεπτομέρεια (Registers).

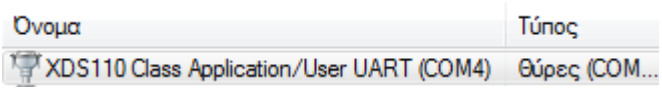
Εικόνα 53. Energia IDE version 1.6.10E18

Δημιουργία απλής εφαρμογής και φόρτωσης της στο αναπτυξιακό MSP-EXP432P401R σε γλώσσα προγραμματισμού C. Η διαδικασία δημιουργίας της εφαρμογής θα αναλυθεί βήμα προς βήμα, καθώς θα γίνει και μια περιήγηση των λειτουργιών του Energia IDE v1.6.10E18.

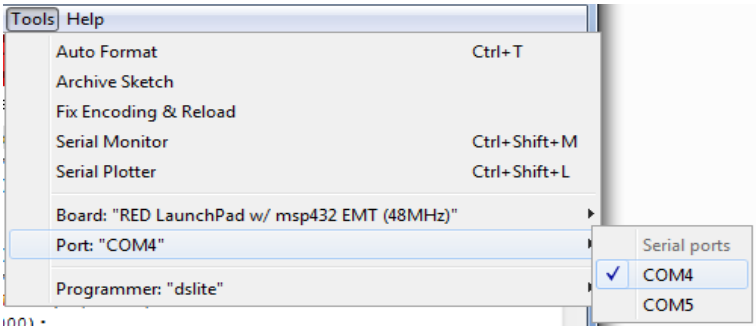
1. Πρώτα συνδέεται μέσω του USB, τον μικροελεγκτή στον υπολογιστή
2. Ανοίγετε το Energia 

3. Στην συνέχεια πηγαίνουμε να διαλέξουμε ποιά πλακέτα αναπτυξιακού χρησιμοποιούμαι Tools (Εργαλεία) > Board (Πλακέτα): " (όνομα αναπτυξιακού) " > λίστα αναπτυξιακών, αν δεν υπάρχει στην λίστα τότε πάμε στο Tools > Board: " (όνομα αναπτυξιακού) " > Board Manager… (Διαχειριστής Πλακετών...) και κάνουμε install (εγκατάσταση) τον κατάλληλο Board Manager

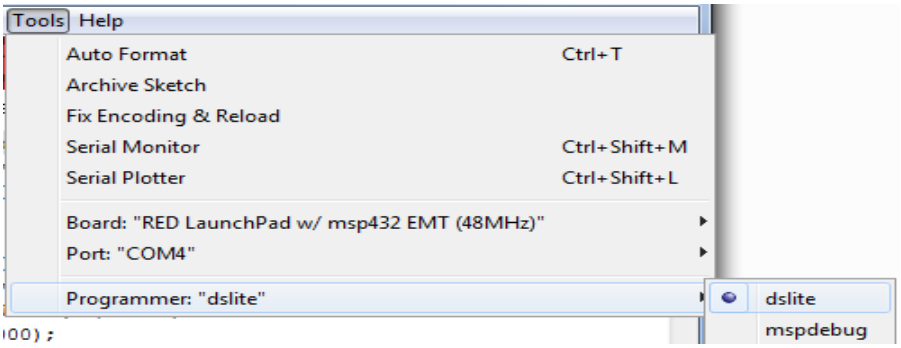


4. Έπειτα κάνουμε τις κατάλληλες ρυθμίσεις για να μπορέσουμε να επικοινωνήσουμε σωστά με τον υπολογιστή, πηγαίνουμε Tools > Port: "COM(νούμερο πόρτας)" και διαλέγουμε την πόρτα που είναι συνδεμένος ο μικροελεγκτής θα το βρούμε με την ονομασία 

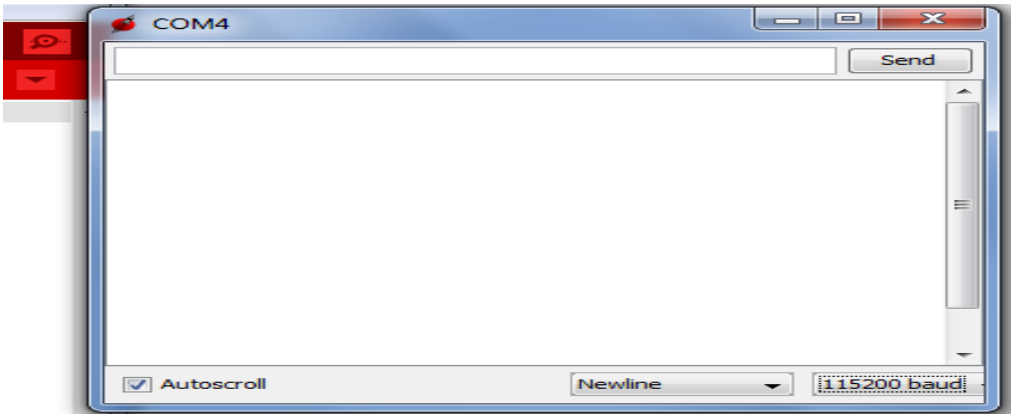
στις συσκευές που είναι συνδεμένες στον υπολογιστή



5. Διαλέγουμε επίσης τον κατάλληλο Programmer (Προγραμματιστή), Tools > Programmer: "dslite"



6. Το Energia έχει 7 βασικά κουμπιά (buttons)
   a. Το κουμπί επικύρωσης (verify)  που κάνει compile το πρόγραμμα
   b. Το κουμπί ανεβάσματος (Upload)  που ανεβάζει το πρόγραμμα στην flash memory, το κάνει αυτόματα και compile αν δεν το έχουμε κάνει εμείς
   c. Το κουμπί δημιουργίας (New)  που μας φτιάχνει νέο πρόγραμμα (New sketch)
   d. Το κουμπί ανοίγματος (Open)  που μας ανοίγει ένα υπάρχων πρόγραμμα
   e. Το κουμπί αποθήκευσης (Save)  που μας αποθηκεύει το τρέχων πρόγραμμα
   f. Το κουμπί παρακολούθησης της σειριακής (Serial Monitor)  μας ανοίγει ένα παράθυρο και μας δίνει την οπτική δυνατότητα παρακολούθησης της σειριακής πόρτας, κάτω δεξιά αυτού του παραθύρου ρυθμίζεται ο ρυθμός μετάδοσης (baud rate)

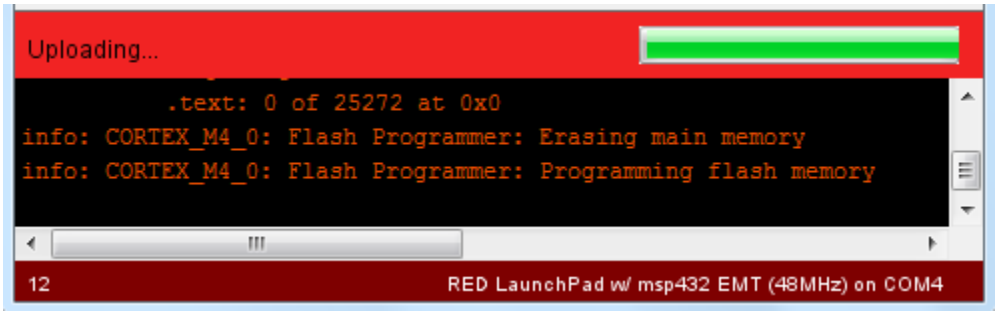


   g. Το κουμπί δημιουργίας νέας καρτέλας (New Tab)  μας δίνει την δυνατότητα να δημιουργήσουμε κι αλλά προγράμματα που να εκτελούνται ταυτόχρονα μεταξύ τους (Multitasking), εκμεταλλεύοντας την τεχνική 3-stage pipeline του Harvard που είδαμε στο Κεφάλαιο 2.
7. Στην συνέχεια για να δημιουργήσουμε ένα νέο πρόγραμμα πατάμε File > New ή πατάμε το κουμπί 
8. Γράφουμε τον κώδικα του προγράμματος μας (externalBlueLedBlink), να αναβοσβήνει ένα Led που είναι συνδεμένο στο pin 40 (P2.7)

Κώδικας Energia σε γλώσσα προγραμματισμού C:

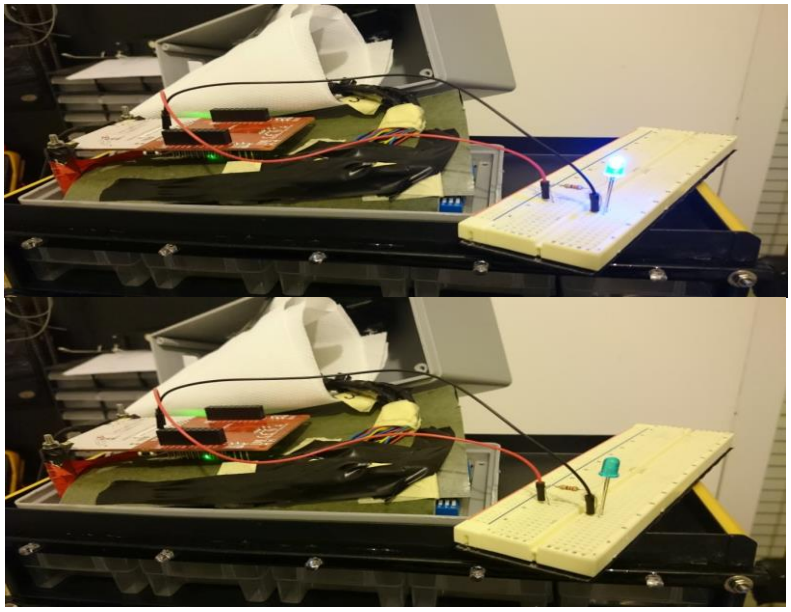
```
void setup() {
  // put your setup code here, to run once:
  pinMode(40, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(40, HIGH);
  delay(1000);
  digitalWrite(40, LOW);
  delay(1000);
}
```

9. Έπειτα πατάμε το κουμπί ανεβάσματος  , μας ζητάει να του δώσουμε όνομα και τοποθεσία αποθήκευσης του προγράμματος, εφόσον γίνει η αποθήκευση, κάνει compile το πρόγραμμα και το φορτώνει στην flash memory του μικροελεγκτή μας, όλη η διαδικασία compile και φορτώματος φαίνεται κάτω στον IDE μας



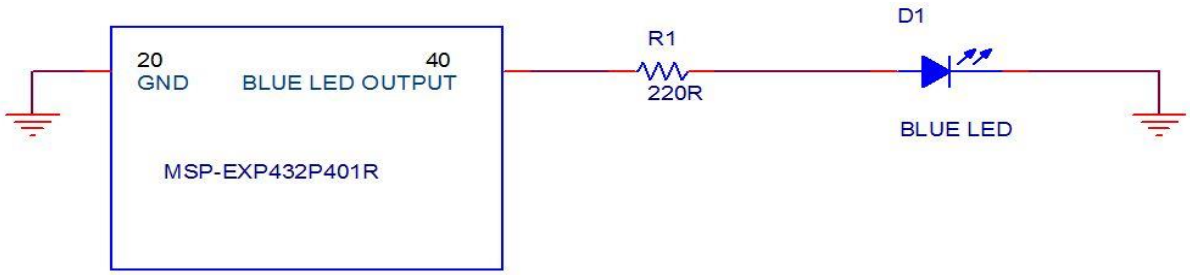
Επίσης όσο φορτώνεται το πρόγραμμα στην μνήμη του μικροελεγκτή ενδεικτικά αναβοσβήνει το κόκκινο LED πάνω στο αναπτυξιακό MSP-EXP432P401R.

Φωτογραφία του προγράμματος που φτιάξαμε με την ονομασία externalBlueLedBlink σε λειτουργία αναβοσβήνοντας ένα μπλε LED.

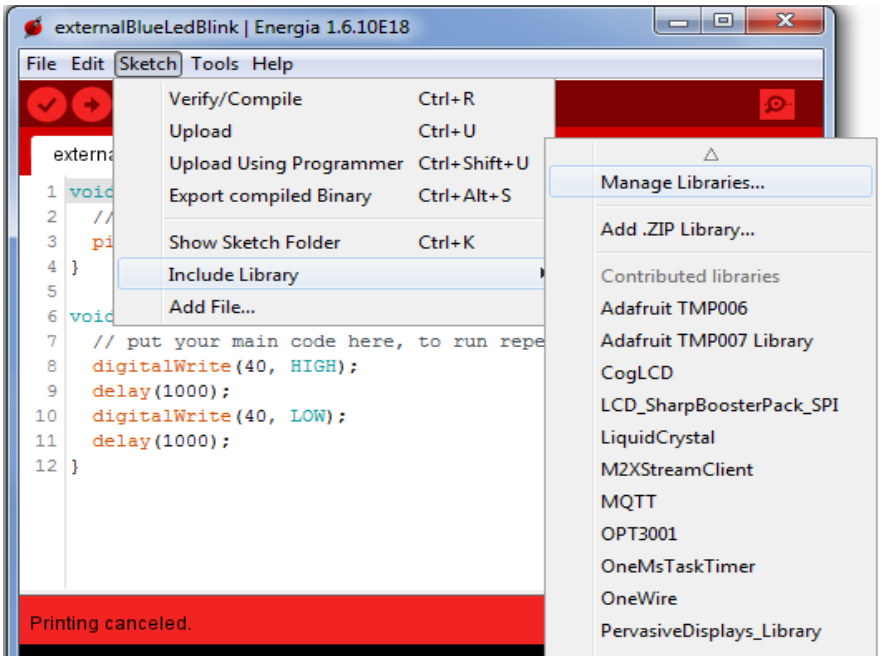


Εικόνα 54. Λειτουργία προγράμματος (Energia Sketch) externalBlueLedBlink

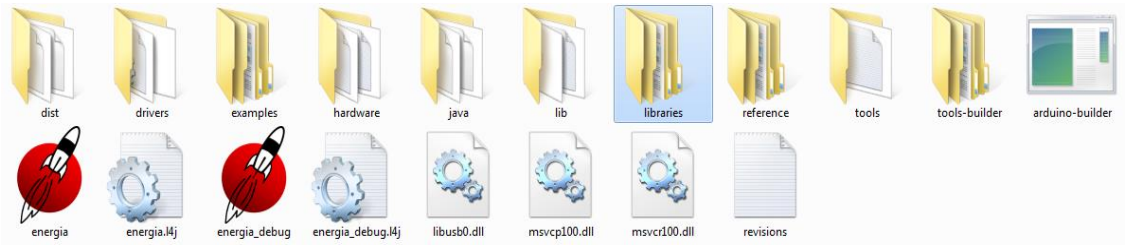
Χρησιμοποιήθηκε το Σχηματικό 1.



Εάν θέλουμε να εισάγουμε μια βιβλιοθήκη στο Energia ακολουθούμε την εξής διαδικασία Sketch (Σχέδιο) > Include Library (Συμπερίληψη Βιβλιοθήκης) > επιλογή βιβλιοθήκης από την λίστα, αν δεν υπάρχει στην λίστα τότε πάμε στο Manage Libraries (διαχείριση βιβλιοθηκών) και την εγκαθιστούμε (install),



αν την έχουμε κατεβάσει την βιβλιοθήκη από το διαδίκτυο μπορούμε να την βάλουμε αμέσως στον φάκελο Libraries του φάκελου εγκατάστασης του Energia IDE



και κανονικά μετά εμφανίζεται στην λίστα των βιβλιοθηκών, αν υπάρχει μια βιβλιοθήκη ήδη στην λίστα απλά γράφουμε #include <(όνομα βιβλιοθήκης).h> ή βιβλιοθήκη του msp δεν φαίνεται στην λίστα αλλά υπάρχει ήδη στο Energia, άρα μπορούμε εύκολα να την εισάγουμε γράφοντας #include "msp.h" ή #include <msp.h> .

Ανάλυση βασικών εντολών του Energia IDE

Εντολές ψηφιακών θυρών I/O

Ο Energia IDE προγραμματίζεται σε γλώσσα προγραμματισμού C και έχει επιπλέον συναρτήσεις / εντολές που μας βοηθάνε στην γρήγορη και αποτελεσματική υλοποίηση προγραμμάτων. Επίσης με την εισαγωγή νέων βιβλιοθηκών, μεγαλώνει η λίστα των εντολών.

Όνομα συνάρτησης:

setup()

Περιγραφή συνάρτησης:

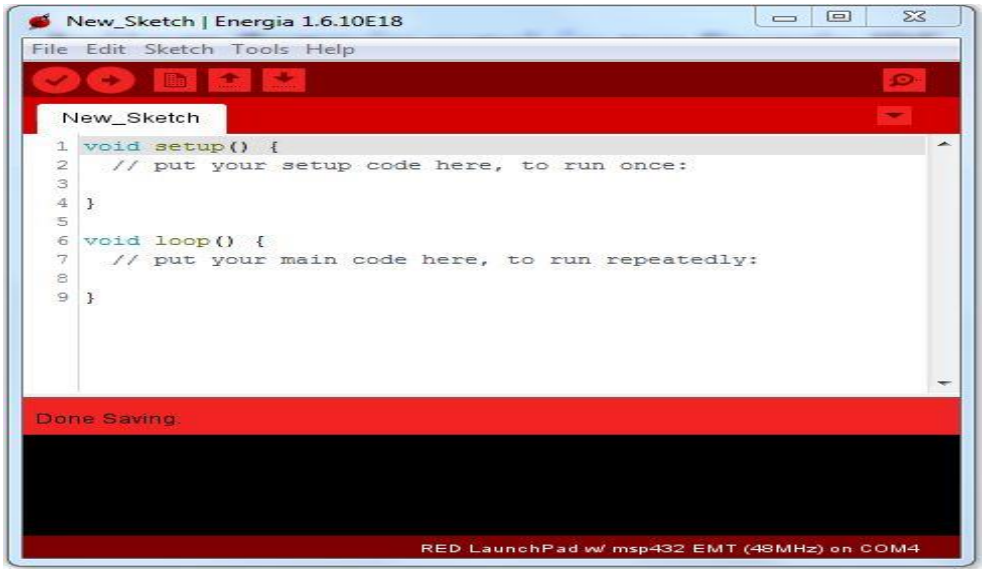
Η συνάρτηση setup() καλείται όταν το πρόγραμμα ξεκινά. Χρησιμοποιείται για την αρχικοποίηση των μεταβλητών (variables), τρόπους λειτουργίας των pin (pin modes), να αρχίσετε να χρησιμοποιείται βιβλιοθήκες, κ.α. Η συνάρτηση setup() τρέχει μόνο μια φορά μετά από κάθε ενεργοποίηση ή επαναφορά του MSP-EXP432P401R.

Όνομα συνάρτησης:

loop()

Περιγραφή συνάρτησης:

Μετά την δημιουργία της συνάρτησης setup(), που αρχικοποιεί και ορίζει τις αρχικές τιμές, η συνάρτηση loop() κάνει ακριβώς αυτό που λέει το όνομα τις, δημιουργεί ένα ατέρμον βρόχο, επιτρέποντας στο πρόγραμμα να αλλάζει και ανταποκρίνεται. Εδώ γράφεται το κυρίως πρόγραμμα μας που θέλουμε να τρέχει συνέχεια.



Εικόνα 55. Συναρτήσεις Setup() και loop()

Όνομα συνάρτησης / εντολής:

pinMode()

Περιγραφή συνάρτησης / εντολής:

Ρυθμίζει το καθορισμένο pin ώστε να συμπεριφέρεται είτε ως είσοδος (input) είτε ως έξοδος (output).

Ο MSP-EXP432P401R έχει την δυνατότητα ενεργοποίησης των εσωτερικών αντιστάσεων pullup ή pulldown με τη λειτουργία (mode) INPUT_PULLUP ή INPUT_PULLDOWN αντίστοιχα, οι pulldown αντιστάσεις δεν λειτουργούν για τους εσωτερικούς διακόπτες του αναπτυξιακού.

Επιπλέον, η λειτουργία INPUT απενεργοποιεί ρητά τα εσωτερικά pullups ή pulldowns.

Σύνταξη εντολής:

pinMode(pin, mode)

Παράμετροι:

pin: Ο αριθμός του pin που θέλουμε να ρυθμίσουμε

mode: INPUT, OUTPUT, INPUT_PULLUP ή INPUT_PULLDOWN

Επιστρέφει:

Τίποτα

Παράδειγμα:

pinMode(40, OUTPUT);

Το pin 40 (P2.7) είναι έξοδος.

Όνομα συνάρτησης / εντολής:

digitalWrite()

Περιγραφή συνάρτησης / εντολής:

Γράφει HIGH (3.3 V) ή LOW (O V , Ground) τιμή (Value) σε ένα ψηφιακό pin.

Σύνταξη εντολής:

digitalWrite(pin, value)

Παράμετροι:

pin: Ο αριθμός του pin που θέλουμε να ρυθμίσουμε

value: HIGH ή LOW

Επιστρέφει:

Τίποτα

Παράδειγμα:

```
digitalWrite(40, HIGH);
```

Το pin 40 έχει δηλωθεί στο setup() σαν έξοδος και η τιμή εξόδου του είναι HIGH (3.3 V), άρα αν συνδέαμε ένα Led στο pin 40 θα άναβε.

Όνομα συνάρτησης / εντολής:

digitalRead()

Περιγραφή συνάρτησης / εντολής:

Διαβάζει την τιμή από ένα καθορισμένο ψηφιακό pin, είτε HIGH είτε LOW.

Σύνταξη εντολής:

```
digitalRead(pin)
```

Παράμετροι:

pin: Ο αριθμός του ψηφιακού pin που θέλουμε να διαβάσουμε

Επιστρέφει:

HIGH (1) ή LOW (0)

Παράδειγμα:

```
int val = digitalRead(40);
```

Διαβάζει το pin 40 και επιστρέφει την τιμή του που είναι είτε HIGH (1) είτε LOW (0) στην ακέραια (int) μεταβλητή με το όνομα val.

Εντολές αναλογικών θυρών I/O και παραγωγή κυματομορφής PWM

Όνομα συνάρτησης / εντολής:

analogReadResolution()

Περιγραφή συνάρτησης / εντολής:

Η εντολή analogReadResolution() ρυθμίζει τα bits της ανάλυσης του μετατροπέα αναλογικού σήματος σε ψηφιακό σήμα [Analog to Digital Converter (A/D)]. Σε προκαθορισμένη ρύθμιση η ανάλυση του A/D είναι 10 bits.

Σύνταξη εντολής:

```
analogReadResolution(bits)
```

Παράμετροι:

bits: καθορίζει την ανάλυση (σε bits) της τιμής που επιστρέφεται από τη λειτουργία της συνάρτησης analogRead (). Τιμή από 1 bit έως 14 bits για τον MSP-EXP432P401R.

Επιστρέφει:

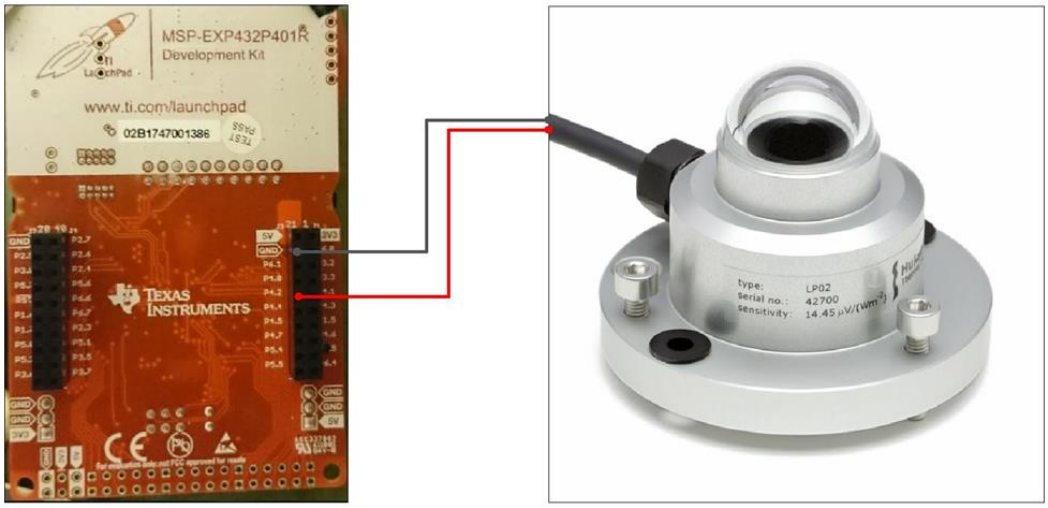
Τίποτα

Παράδειγμα:

Η εντολή analogReadResolution(14) χρησιμοποιήθηκε για να ρυθμίσουμε την ανάλυση του A/D στα 14 bits, ώστε να έχουμε την δυνατότητα να διαβάσουμε με την analogRead() τις μετρήσεις ενός πυρανόμετρου.

Το πυρανόμετρο μας δίνει μετρήσεις της τάξης των 0 mV έως 10 mV και θέλαμε ανάλυση ±0.2 mV, άρα όταν εμείς ρυθμίσουμε την ανάλυση στα 14 bits με αναφορά τα 3.3 V, η ανάλυση γίνεται 0.2 mV/unit (βλέπε περιγραφή εντολής analogRead() που ακολουθεί).

Ακολουθεί η υλοποίηση του κυκλώματος του πυρανόμετρου για να διαβαστούν οι μετρήσεις του από τον μικροελεγκτή μας. Το πυρανόμετρο συνδέεται στην αναλογική θύρα 25 (P4.2)



Εικόνα 56. Απευθείας σύνδεση πυρανόμετρου με τον MSP-EXP432P401R

Σημειώσεις και προειδοποιήσεις:

Αν γράψεις αριθμό bits παραπάνω από αυτό που υποστηρίζει το αναπτυξιακό που χρησιμοποιείται, συμπληρώνει τα επιπλέον bits με μηδενικά.

Όνομα συνάρτησης / εντολής:

analogRead()

Περιγραφή συνάρτησης / εντολής:

Διαβάζει την τιμή από το καθορισμένο αναλογικό pin. Η μέγιστη ανάλυση του A/D στον MSP-EXP432P401R είναι 14 bits αν την ρυθμίσουμε με την εντολή analogReadResolution(), αλλιώς η ανάλυση είναι προκαθορισμένη στα 10 bits. Αυτό σημαίνει αν είναι στα 10 bits η ανάλυση, ότι θα αντιστοιχίσει τάσεις εισόδου μεταξύ 0 και 3.3 βολτ σε ακέραιες τιμές μεταξύ 0 και 1023 γιατί $analogReadValue = 2^{Resolution\ bits} - 1 = 2^{10} - 1 = 1023\ units$ (αν ήταν 14 bits θα ήταν μεταξύ του 0 και 16383 γιατί $analogReadValue = 2^{14} - 1 = 16383$). Αυτό σημαίνει ότι μας δίνει μια ανάλυση μεταξύ των μετρήσεων για τα 10 bits: $resolution\ \left(\frac{Volt}{Unit}\right) = \frac{Analog\ Volt\ Reference}{2^{Resolution\ bits}} = \frac{3.3\,V}{2^{10}} = 0.0032\ V/Unit$ ή 3.2 mV/Unit, 3.2 mV ανά μονάδα.

Στην μέγιστη ρύθμιση ανάλυσης στα 14 bits θα μας δίνει:

$resolution\ \left(\frac{Volt}{Unit}\right) = \frac{Analog\ Volt\ Reference}{2^{Resolution\ bits}} = \frac{3.3\,V}{2^{14}} = 0.00020142\ V/Unit$ ή 0.20142 mV/Unit

Σύνταξη εντολής:

analogRead(pin)

Παράμετροι:

pin: ο αριθμός του αναλογικού pin εισόδου προς ανάγνωση

Επιστρέφει:

Στην προκαθορισμένη ρύθμιση των 10 bits επιστρέφει ακέραιο από 0 έως 1023 [int(0 to 1023)], στην μέγιστη ρύθμιση των 14 bits επιστρέφει ακέραιο από 0 έως 16383 [int(0 to 16383)].

Παράδειγμα:

int analogVal = analogRead(23);

Διαβάζει την τιμή του αναλογικού pin 23 (P6.1) και επιστρέφει την τιμή του στην ακέραια μεταβλητή analogVal με εύρος τιμών αναλόγως τα bits της ανάλυσης.

Αν θέλουμε να δούμε το αποτέλεσμα στην πραγματική τάση σε βόλτ που διαβάζει η είσοδος ο μαθηματικός τύπος είναι:

$$Volt = \frac{analogRead(pin) \times analogVoltReference}{2^{Resolution\ bits} - 1}$$

π.χ. για 10 bits ανάλυση και τιμή του αναλογικού pin ίσο με 500 (από την analogRead()) η τάση που διαβάζουμε θα είναι:

$$Volt = \frac{analogRead(pin) \times analogVoltReference}{2^{Resolution\ bits} - 1} = \frac{500 \times 3.3\ V}{2^{10} - 1} = 1.61\ V$$

Όνομα συνάρτησης / εντολής:

analogWrite()

Περιγραφή συνάρτησης / εντολής:

Γράφει μια κυματομορφή PWM σε ένα pin. Μετά το κάλεσμα της η analogWrite() παράγει ένα σταθερό τετραγωνικό παλμό συγκεκριμένου duty cycle, μέχρι να την καλέσουμε ξανά την analogWrite(), ή την digitalRead(), ή την digitalWrite() στο ίδιο pin. Η συχνότητα του PWM σήματος είναι 490 Hz.

Δεν χρειάζεται να καλέσεις την συνάρτηση pinMode() για να ορίσεις το pin σαν έξοδο πριν καλέσεις την συνάρτηση analogWrite().

Σύνταξη εντολής:

analogWrite(pin, value)

Παράμετροι:

pin: Το pin που θα γράψει σε αυτό.

value: Το duty cycle (0-255): μεταξύ του 0 (πάντα σβηστό(off)) και 255 (πάντα ανοιχτό(on)).

Επιτρέπονται οι τύποι δεδομένων: int.

Επιστρέφει:

Τίποτα

Παράδειγμα:

analogWrite(40, 85);

Γράφει στο pin 40 (P2.7) μια PWM κυματομορφή με duty cycle ίσο με 33.3%

Το duty cycle υπολογίζεται από τον μαθηματικό τύπο:

$$duty\ cycle = \frac{85}{255} \times 100 = 33.3\%$$

Εντολές καθυστέρησης και μέτρησης του χρόνου

Όνομα συνάρτησης / εντολής:

delay()

Περιγραφή συνάρτησης / εντολής:

Σταματάει το πρόγραμμα για το χρονικό διάστημα (σε milliseconds) που ορίζεται ως παράμετρος.

Σύνταξη εντολής:

delay(ms)

Παράμετροι:

ms: Ο αριθμός των milliseconds που θα σταματήσει το πρόγραμμα (Τύπος: unsigned long)

Επιστρέφει:

Τίποτα

Παράδειγμα:

delay(1000);

Καθυστερεί για 1000 milliseconds το πρόγραμμα δηλαδή για 1 δευτερόλεπτο.

Όνομα συνάρτησης / εντολής:

delayMicroseconds()

Περιγραφή συνάρτησης / εντολής:

Σταματάει το πρόγραμμα για το χρονικό διάστημα (σε microseconds) που ορίζεται ως παράμετρος.

Σύνταξη εντολής:

delayMicroseconds(us)

Παράμετροι:

us: Ο αριθμός των microseconds που θα σταματήσει το πρόγραμμα (Τύπος: unsigned int)

Επιστρέφει:

Τίποτα

Παράδειγμα:

delayMicroseconds(1000);

Καθυστερεί για 1000 microseconds το πρόγραμμα δηλαδή για 1 millisecond.

Όνομα συνάρτησης / εντολής:

millis()

Περιγραφή συνάρτησης / εντολής:

Επιστρέφει τον αριθμό των milliseconds από τότε που ο μικροελεγκτής ξεκίνησε να τρέχει το τρέχων πρόγραμμα. Είτε δηλαδή μπήκε στην τροφοδοσία, είτε έγινε Reset.

Σύνταξη εντολής:

millis()

Παράμετροι:

Τίποτα

Επιστρέφει:

Επιστρέφει τον αριθμό των milliseconds από τότε που ο μικροελεγκτής ξεκίνησε να τρέχει το τρέχων πρόγραμμα. (Τύπος: unsigned long)

Παράδειγμα:

```
unsigned long time = millis();
```

Επιστρέφει τον χρόνο σε milliseconds μέχρι εκείνη την στιγμή, π.χ. αν έχουμε συνδέσει τον μικροελεγκτή μας 10 δευτερόλεπτα από την στιγμή που παίρνουμε την μέτρηση, θα μας δώσει την τιμή 10000 milliseconds στην unsigned long μεταβλητή με το όνομα time που ισούται με τα 10 δευτερόλεπτα.

Όνομα συνάρτησης / εντολής:

micros()

Περιγραφή συνάρτησης / εντολής:

Επιστρέφει τον αριθμό των microseconds από τότε που ο μικροελεγκτής ξεκίνησε να τρέχει το τρέχων πρόγραμμα. Είτε δηλαδή μπήκε στην τροφοδοσία, είτε έγινε Reset.

Σύνταξη εντολής:

micros()

Παράμετροι:

Τίποτα

Επιστρέφει:

Επιστρέφει τον αριθμό των microseconds από τότε που ο μικροελεγκτής ξεκίνησε να τρέχει το τρέχων πρόγραμμα. (Τύπος: unsigned long)

Παράδειγμα:

```
unsigned long time = micros();
```

Επιστρέφει τον χρόνο σε microseconds μέχρι εκείνη την στιγμή, π.χ. αν έχουμε συνδέσει τον μικροελεγκτή μας 1 δευτερόλεπτο από την στιγμή που παίρνουμε την μέτρηση, θα μας δώσει την τιμή 1000000 microseconds στην unsigned long μεταβλητή με το όνομα time που ισούται με το 1 δευτερόλεπτο.

Εντολές σειριακής θύρας

Όνομα συνάρτησης / εντολής:

Serial.begin()

Περιγραφή συνάρτησης / εντολής:

Ορίζει τον ρυθμό δεδομένων σε bits ανά δευτερόλεπτο [bits per second (bps)] (baud) για τη μετάδοση σειριακών δεδομένων

Σύνταξη εντολής:

Serial.begin(speed)

Παράμετροι:

speed: ταχύτητα σε bits ανά δευτερόλεπτο (baud), (τύπος: long), η ταχύτητα μπορεί να πάρει τιμές όπως, 9600 baud, 19200 baud, 38400 baud, 57600 baud, 74880 baud, 115200 baud κ.α.

Επιστρέφει:

Τίποτα

Παράδειγμα:

Serial.begin(115200);

Ανοίγει την σειριακή θύρα, ρυθμίζει τον ρυθμό δεδομένων στα 115200 baud (bps).

Όνομα συνάρτησης / εντολής:

Serial.end()

Περιγραφή συνάρτησης / εντολής:

Απενεργοποιεί τη σειριακή επικοινωνία, επιτρέποντας στους ακροδέκτες (pins) RX και TX να χρησιμοποιούνται για γενική είσοδο και έξοδο. Για να ενεργοποιήσετε εκ νέου τη σειριακή επικοινωνία, καλέστε την εντολή Serial.begin ().

Σύνταξη εντολής:

Serial.end()

Παράμετροι:

Τίποτα

Επιστρέφει:

Τίποτα

Παράδειγμα:

Serial.end();

Απενεργοποιεί τη σειριακή επικοινωνία.

Όνομα συνάρτησης / εντολής:

Serial.print()

Περιγραφή συνάρτησης / εντολής:

Εκτυπώνει δεδομένα στη σειριακή θύρα, ως ASCII κείμενο αναγνώσιμο από τον άνθρωπο.

Σύνταξη εντολής:

Serial.print(val)

Serial.print(val, format)

Παράμετροι:

val: Η τιμή για εκτύπωση, υποστηρίζει οποιοδήποτε τύπο δεδομένων

format: καθορίζει τη βάση αριθμών (για τους ακέραιους (integral) τύπους δεδομένων) ή τον αριθμό των δεκαδικών ψηφίων (για τύπους κυμαινόμενων σημείων (floating point))

Επιστρέφει:

Τίποτα

Παράδειγμα:

Serial.print("Hello world");

Εμφανίζει / εκτυπώνει στο Serial Monitor την έκφραση: Hello world συνέχεια στην ίδια γραμμή.

Όνομα συνάρτησης / εντολής:

Serial.println()

Περιγραφή συνάρτησης / εντολής:

Εκτυπώνει δεδομένα στη σειριακή θύρα, ως ASCII κείμενο αναγνώσιμο από τον άνθρωπο. Είναι ακριβώς ίδια εντολή με την Serial.print() με μόνη διαφορά ότι έχει ένα παραπάνω χαρακτήρα νέας γραμμής (ASCII 10, ή '\ n').

Σύνταξη εντολής:

Serial.println(val)

Serial.println(val, format)

Παράμετροι:

val: Η τιμή για εκτύπωση, υποστηρίζει οποιοδήποτε τύπο δεδομένων

format: καθορίζει τη βάση αριθμών (για τους ακέραιους (integral) τύπους δεδομένων) ή τον αριθμό των δεκαδικών ψηφίων (για τύπους κυμαινόμενων σημείων (floating point))

Επιστρέφει:

Τίποτα

Παράδειγμα:

```
Serial.println("Hello  world");
```

Εμφανίζει / εκτυπώνει στο Serial Monitor την έκφραση: Hello world και αλλάζει κάθε φορά γραμμή και ξανά εκτυπώνει.

Άλλο παράδειγμα:

```
int  a = 5;
Serial.println(a);
```

Εκτυπώνει τον ακέραιο αριθμό 5 στο Serial Monitor συνέχεια και αλλάζει γραμμή κάθε φορά και ξανά εκτυπώνει.

Εντολές για διακοπές (Interrupts)

Όνομα συνάρτησης / εντολής:

noInterrupts()

Περιγραφή συνάρτησης / εντολής:

Γενική απενεργοποίηση των interrupts (για ενεργοποίηση τους ξανά γίνεται με την εντολή interrupts())

Σύνταξη εντολής:

noInterrupts()

Παράμετροι:

Τίποτα

Επιστρέφει:

Τίποτα

Παράδειγμα:

```
noInterrupts();
```

Απενεργοποιεί τα interrupts τώρα μπορεί να γραφτεί από κάτω του κρίσιμος, χρονικά ευαίσθητος κώδικας που δεν πρέπει να διακοπεί η λειτουργία του.

Όνομα συνάρτησης / εντολής:

interrupts()

Περιγραφή συνάρτησης / εντολής:

Γενική ενεργοποίηση των interrupts

Σύνταξη εντολής:

interrupts()

Παράμετροι:

Τίποτα

Επιστρέφει:

Τίποτα

Παράδειγμα:

interrupts();

Ενεργοποιεί τα interrupts, τώρα σε οποιαδήποτε κρίσιμη στιγμή ενεργοποιούνται κανονικά οι διακοπές (interrupts) που έχουμε ορίσει ή που είναι ορισμένες αυτόματα από το σύστημα.

Όνομα συνάρτησης / εντολής:

attachInterrupt()

Περιγραφή συνάρτησης / εντολής:

Καθορίζει μια λειτουργία που καλείται όταν εμφανιστεί μια εξωτερική διακοπή.

Σύνταξη εντολής:

attachInterrupt(interruptPin, function, mode)

Παράμετροι:

interruptPin: Ο αριθμός του pin που θα ρυθμιστή ως διακοπή (interrupt) (Τύπος: int)

function: το όνομα της συνάρτησης που καλείται όταν υπάρξει διακοπή, αυτή η διακοπή δεν παίρνει παραμέτρους ούτε επιστρέφει κάτι. Αυτή η συνάρτηση αναφέρεται και σαν ρουτίνα εξυπηρέτησης διακοπής [interrupt service routine (ISR)].

mode: ορίζει πότε πρέπει να ενεργοποιηθεί η διακοπή. Τέσσερις σταθερές είναι προκαθορισμένες ως έγκυρες τιμές:

LOW για να ενεργοποιήσει το interrupt, κάθε φορά που το pin είναι σε low κατάσταση.

CHANGE για να ενεργοποιήσει το interrupt, όποτε το pin αλλάζει τιμή.

RISING για να ενεργοποιήσει το interrupt, όταν το pin πάει από κατάσταση low σε high δηλαδή σε κάθε ανερχόμενο μέτωπο.

FALLING για να ενεργοποιήσει το interrupt, όταν το pin πάει από κατάσταση high σε low δηλαδή σε κάθε κατερχόμενο μέτωπο.

Επιστρέφει:

Τίποτα

Παράδειγμα:

attachInterrupt(40, blink, RISING);

Ενεργοποιεί την ISR με το όνομα blink όταν το pin 40 δεκτή ένα σήμα από low σε high.

Σημειώσεις και προειδοποιήσεις:

Για να λειτουργήσουν σωστά τα εξωτερικά interrupts πρέπει να ενεργοποιήσουμε τις εσωτερικές pullup ή pulldown αντιστάσεις ή να βάλουμε εξωτερικές, για να μην "αιωρείται" το pin (δηλαδή να μην γνωρίζομαι σε τη κατάσταση βρίσκεται το pin Low ή High).

Όνομα συνάρτησης / εντολής:

detachInterrupt()

Περιγραφή συνάρτησης / εντολής:

Απενεργοποιεί τη δεδομένη διακοπή (disable interrupt).

Σύνταξη εντολής:

detachInterrupt(interruptPin)

Παράμετροι:

interruptPin: Ο αριθμός του pin που θα απενεργοποιηθεί η διακοπή (interrupt)

Επιστρέφει:

Τίποτα

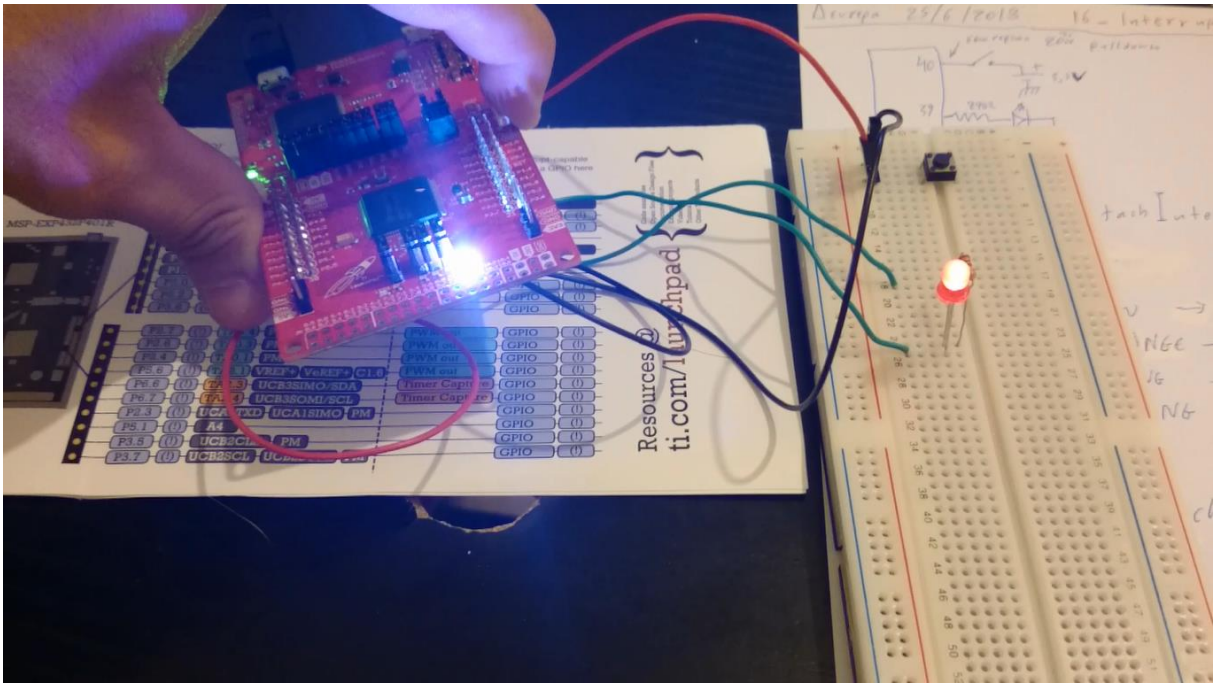
Παράδειγμα:

detachInterrupt(40);

Απενεργοποιεί την διακοπή που υπάρχει στο pin 40 (πρώτα την έχουμε ενεργοποιήσει την διακοπή με την εντολή attachInterrupt() στο pin 40, εφόσον απενεργοποιηθεί με την χρήση της εντολής detachInterrupt() το εξωτερικό interrupt στο pin 40, αν θέλουμε να το ενεργοποιήσουμε ξανά κάνουμε πάλι την χρήση της εντολής attachInterrupt()).

Εφαρμογή Interrupts

Ακολουθούν φωτογραφίες μιας απλής Interrupt εφαρμογής, που συνδυάζει όλες τις παραπάνω εντολές διακοπών, καθώς η υλοποίηση της σε Breadboard.



Η εφαρμογή όταν ξεκινάει ανάβει το εσωτερικό πράσινο Led, έπειτα ενεργοποιεί τρείς διαφορετικές ρουτίνες εξυπηρέτησης διακοπών αναλόγως ποιό κουμπί πατηθεί:

- Όταν **<u>πατηθεί</u>** το εσωτερικό κουμπί PUSH2 ενεργοποιείται η ρουτίνα που ονομάσαμε blink και εναλλάσσει την κατάσταση του εσωτερικού πράσινου led και κάνει High ένα flag. Εκτός ρουτίνας το flag με την σειρά του προσθέτει +1 σε έναν counter, αν ο counter γίνει 2 ανάβει το κόκκινο εσωτερικό led αν γίνει 3 ανάβει και το μπλέ εσωτερικό led, αν γίνει 5 βγάζει το interrupt από το εσωτερικό κουμπί PUSH2, όταν γίνει 10 ο counter, ενεργοποιείται ξανά το Interrupt στο κουμπί PUSH2, επίσης όταν ο counter γίνει 10 προστίθενται σε αυτόν αλλά 10 με μια for.
- Όταν **<u>αφήνουμε</u>** το εσωτερικό κουμπί PUSH1 ενεργοποιείται η ρουτίνα που ονομάσαμε blink2 και εναλλάσσει την κατάσταση του εσωτερικού πράσινου led και κάνει High ένα flag, ακριβώς το ίδιο με την ISR blink.
- Όταν **<u>αφήνουμε</u>** το εξωτερικό κουμπί που είναι συνδεμένο στην θέση 40 του μικροελεγκτή, (το κουμπί βρίσκεται στο Breadboard μαζί με ένα εξωτερικό κόκκινο led) τότε ενεργοποιείται η ISR με το όνομα blink3 η οποία σβήνει το κόκκινο εσωτερικό και το μπλε εσωτερικό led καθώς μηδενίζει τον counter και αλλάζει την κατάσταση του κόκκινου εξωτερικού led που βρίσκεται στο breadboard.

Η εφαρμογή είναι μόνο για έλεγχο των εντολών των διακοπών, ο έλεγχος τιμών του counter γίνεται μέσω του Serial Monitor.

_16_Interrupts | Energia 1.6.10E18

File Edit Sketch Tools Help

_16_Interrupts §

```
  pinMode(40, INPUT_PULLDOWN);

  noInterrupts();
  //attachInterrupt(interrupt, function, mode)

  attachInterrupt(PUSH1, blink2, RISING); // Energopoihsei tou Interrupt otan afhno to koumpi 1
  attachInterrupt(PUSH2, blink, FALLING); // Energopoihsei tou Interrupt otan patao to koumpi 2
  attachInterrupt(40, blink3, FALLING); // Energopoihsei tou Interrupt otan afhno to koumpi sto BREADBOARD

  interrupts();
}

void loop()
{
  digitalWrite(GREEN_LED, state);
  if (flag)
  {
    count++;
    Serial.println(count);
    flag = LOW;
  }

  switch (count)
  {
    case 2: digitalWrite(RED_LED, HIGH); break;
    case 3: digitalWrite(BLUE_LED, HIGH); break;
  }
```

No changes necessary for Auto Format.

_16_Interrupts | Energia 1.6.10E18

File Edit Sketch Tools Help

_16_Interrupts §

```
volatile int state = HIGH;
volatile int flag = HIGH;
volatile int breadBoardLedState = LOW;
int count = 0;

void blink2(void);

void setup()
{
  Serial.begin(9600);

  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);
  pinMode(39, OUTPUT);

  digitalWrite(GREEN_LED, state);
  digitalWrite(RED_LED, LOW);
  digitalWrite(BLUE_LED, LOW);

  //Energopoioume ta eswterika pull up

  pinMode(PUSH1, INPUT_PULLUP);
  pinMode(PUSH2, INPUT_PULLUP);
  pinMode(40, INPUT_PULLDOWN);

  noInterrupts();
```

No changes necessary for Auto Format.

_16_Interrupts | Energia 1.6.10E18

File Edit Sketch Tools Help

_16_Interrupts §

```
    case 3: digitalWrite(BLUE_LED, HIGH); break;
  }

  if (count == 5)
  {
    detachInterrupt(PUSH2);
    Serial.println("Apenergopoihsh tou Interrupt sto Koumpi 2");
  }

  if (count == 10)
  {
    for (int i = 0; i < 10; i++)
    {
      count++;
    }
    noInterrupts();
    attachInterrupt(PUSH2, blink, FALLING); // epanenergopoihsei tou Interrupt PUSH2
    Serial.println("Energopoihsh tou Interrupt sto Koumpi 2");
    interrupts();
  }
}

void blink() //Energopoieitai apo ton eswteriko diakoptha 2
{
  state = !state;
  flag = HIGH;
```

No changes necessary for Auto Format.

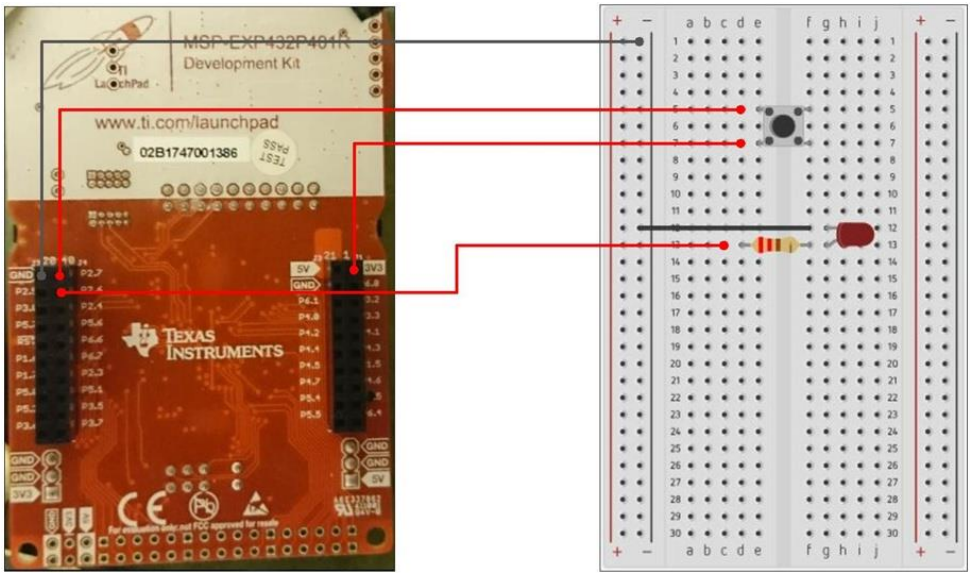
_16_Interrupts | Energia 1.6.10E18

File Edit Sketch Tools Help

_16_Interrupts §

```
    noInterrupts();
    attachInterrupt(PUSH2, blink, FALLING); // epanenergopoihsei tou Interrupt PUSH2
    Serial.println("Energopoihsh tou Interrupt sto Koumpi 2");
    interrupts();
  }
}

void blink() //Energopoieitai apo ton eswteriko diakoptha 2
{
  state = !state;
  flag = HIGH;
}

void blink2() //Energopoieitai apo ton eswteriko diakoptha 1
{
  state = !state;
  flag = HIGH;
}

void blink3() //Energopoieitai apo ton e3wteriko diakopth sto pin 40
{
  digitalWrite(RED_LED, LOW);
  digitalWrite(BLUE_LED, LOW);
  breadBoardLedState = !breadBoardLedState;
  digitalWrite(39, breadBoardLedState);
  count = 0;
  Serial.println(count);
}
```

No changes necessary for Auto Format.



Breadboard 1. Υλοποίηση εφαρμογής Interrupts

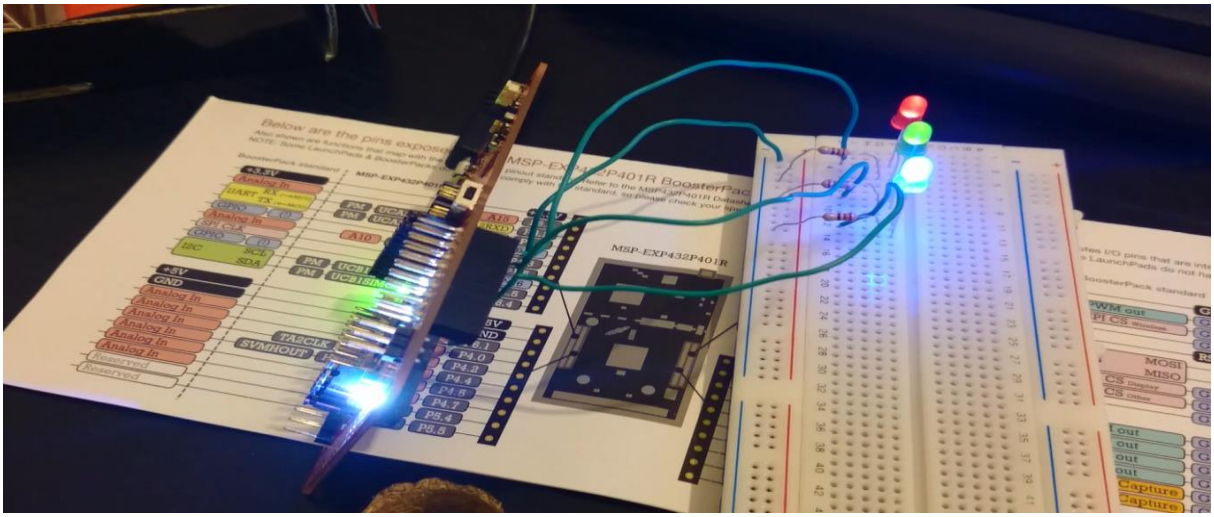
Multitasking

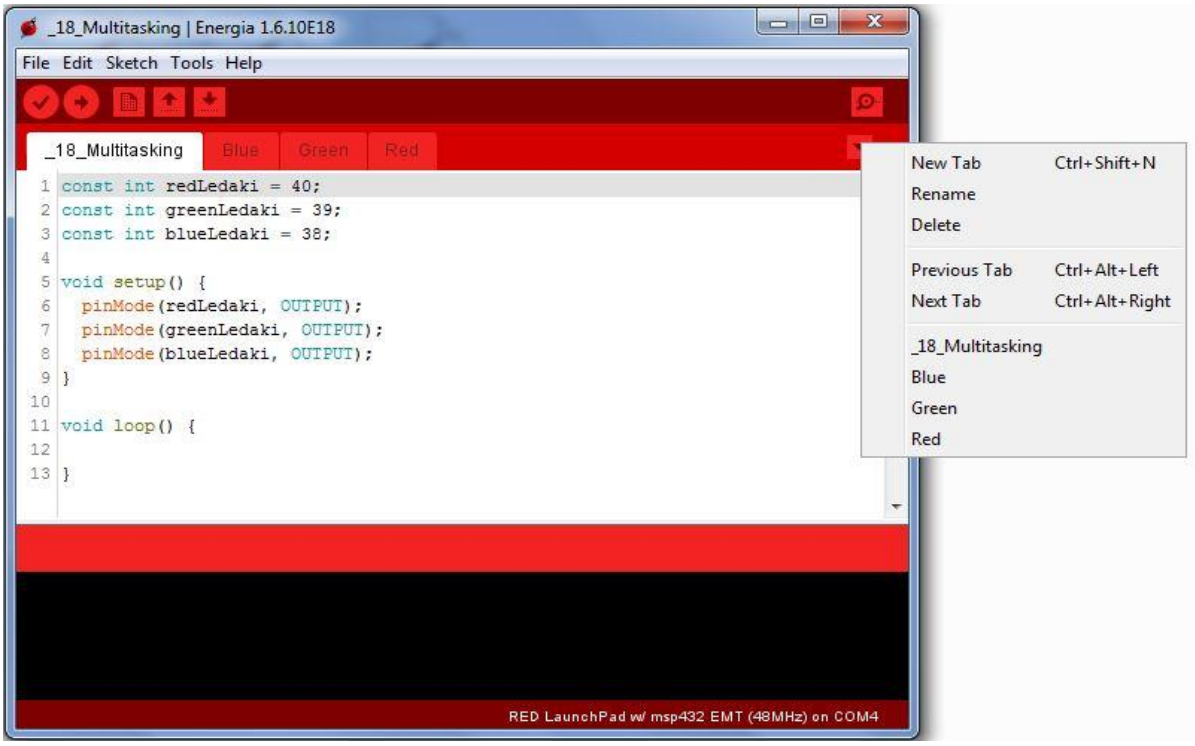
Το Energia IDE μας δίνει την δυνατότητα multitasking, με την δυνατότητα δημιουργίας νέων καρτελών από το κουμπί  .

Το μόνο που πρέπει να προσέξουμε είναι η σύνταξη των συναρτήσεων, setup() και loop(). Στην αρχική καρτέλα μπορούν το setup() και η loop να παραμείνουν έτσι, αλλά κάθε επόμενη καρτέλα πρέπει να έχει την λέξη κλειδί setup+όνομα νέας καρτέλας() και loop+όνομα νέας καρτέλας().

Εφαρμογή Multitasking

Ακολουθούν φωτογραφίες μιας απλής multitasking εφαρμογής για να δούμε την σωστή σύνταξη των setup() και των loop(), καθώς η υλοποίηση της σε Breadboard.





_18_Multitasking - Blue.ino | Energia 1.6.10E18

File Edit Sketch Tools Help

_18_Multitasking | Blue § | Green | Red

```
1  #undef LED //sbhnei tin palia staθera gia na kanei tin nea define
2  #define LED BLUE_LED
3
4  void setupBlue() {
5    pinMode(LED, OUTPUT);
6  }
7
8  void loopBlue() {
9    digitalWrite(LED, HIGH);
10   digitalWrite(blueLedaki, HIGH);
11   delay(1000);
12   digitalWrite(LED, LOW);
13   digitalWrite(blueLedaki, LOW);
14   delay(1000);
15 }
```

1 RED LaunchPad w/ msp432 EMT (48MHz) on COM4

_18_Multitasking - Green.ino | Energia 1.6.10E18

File Edit Sketch Tools Help

_18_Multitasking | Blue | Green | Red

```
1  #undef LED
2  #define LED GREEN_LED
3
4  void setupGreen() {
5    pinMode(LED, OUTPUT);
6  }
7
8  void loopGreen() {
9    digitalWrite(LED, HIGH);
10   digitalWrite(greenLedaki, HIGH);
11   delay(500);
12   digitalWrite(LED, LOW);
13   digitalWrite(greenLedaki, LOW);
14   delay(500);
15 }
```

RED LaunchPad w/ msp432 EMT (48MHz) on COM4

_18_Multitasking - Red.ino | Energia 1.6.10E18

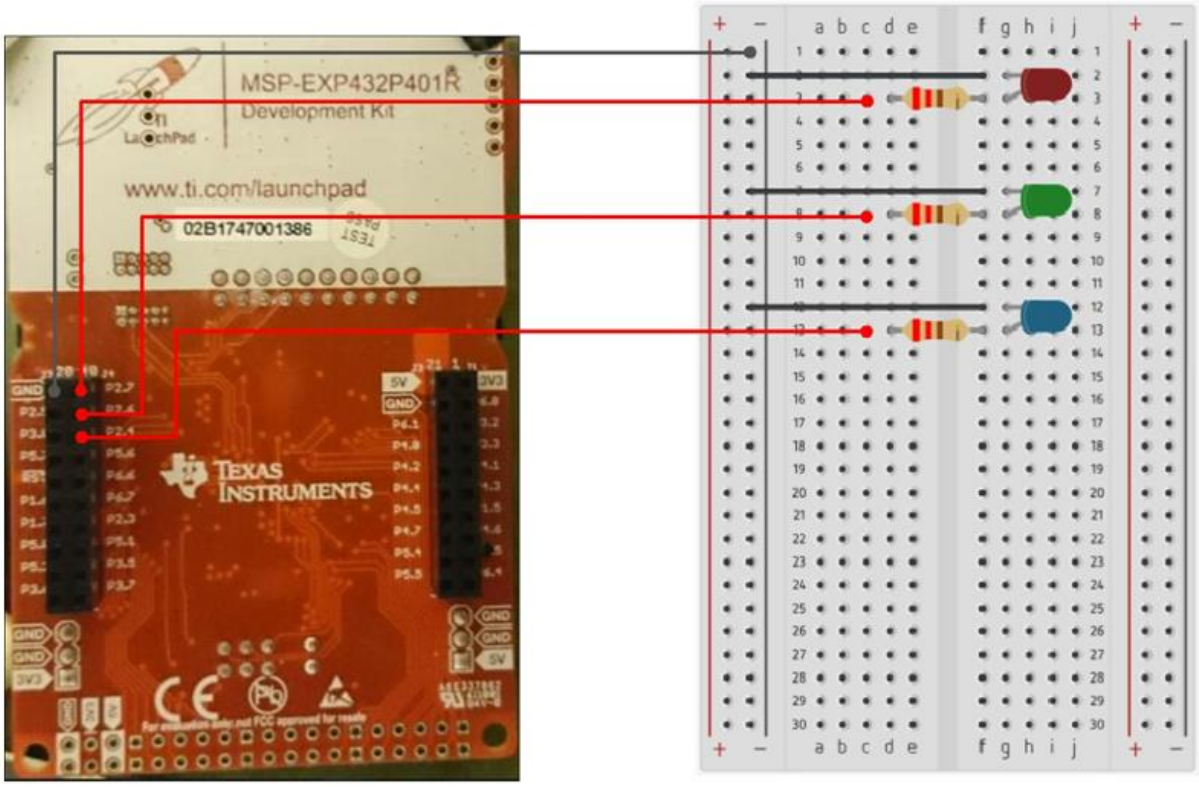
File Edit Sketch Tools Help

_18_Multitasking | Blue | Green | Red

```
#undef LED
#define LED RED_LED

void setupRed() {
  pinMode(LED, OUTPUT);
}

void loopRed() {
  digitalWrite(LED, HIGH);
  digitalWrite(redLedaki, HIGH);
  delay(100);
  digitalWrite(LED, LOW);
  digitalWrite(redLedaki, LOW);
  delay(100);
}
```

RED LaunchPad w/ msp432 EMT (48MHz) on COM4

Η παραπάνω εφαρμογή απλώς αναβοσβήνει το εσωτερικό RGB LED του MSP-EXP432P401R και 3 εξωτερικά LED στις θέσεις pin 40, 39 και 38.



Breadboard 2. Υλοποίηση εφαρμογής Multitasking

Εντολές για οθόνη υγρών κρυστάλλων [Liquid Crystal Display (LCD)]

Πρώτο βήμα για τον έλεγχο της LCD οθόνης, είναι να γίνει η εισαγωγή της κατάλληλης βιβλιοθήκης, που διαθέτει όλες εκείνες τις εντολές που θα μας βοηθήσουν εύκολα να την προγραμματίσουμε. Άρα εισάγουμε την βιβλιοθήκη με το όνομα LiquidCrystal.h με έναν από τους τρόπους που περιγράφτηκαν π.χ. #include <LiquidCrystal.h> εφόσον έχει εγκατασταθεί από το Manage Libraries.

Έπειτα φτιάχνουμε μια μεταβλητή (αντικείμενο) με τύπο LiquidCrystal με τους κατάλληλους παραμέτρους.

Αυτή η βιβλιοθήκη επιτρέπει συγκεκριμένα, τον έλεγχο LCD οθόνης πού είναι βασισμένη στο chipset Hitachi HD44780 (ή κάποιο άλλο συμβατό), το οποίο βρίσκεται στις περισσότερες LCD οθόνες που είναι βασισμένες στο κείμενο. Η βιβλιοθήκη λειτουργεί σε 4 ή 8 bit mode (δηλ. Χρησιμοποιεί 4 ή 8 γραμμές δεδομένων επιπλέον των rs, enable και προαιρετικά, τις γραμμές ελέγχου rw).

Σύνταξη:

LiquidCrystal όνομα μεταβλητής(rs, enable, d4, d5, d6, d7)

LiquidCrystal όνομα μεταβλητής(rs, rw, enable, d4, d5, d6, d7)

LiquidCrystal όνομα μεταβλητής(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)

LiquidCrystal όνομα μεταβλητής(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

Παράμετροι:

rs: Ο αριθμός του pin στον μικροελεγκτή που θα συνδεθεί το Register Select (RS) της LCD οθόνης, το rs είναι υπεύθυνο για το που θα εμφανίζεται ο χαρακτήρας στην οθόνη.

rw: Ο αριθμός του pin στον μικροελεγκτή που θα συνδεθεί το Read/Write (R/W) της LCD οθόνης (προαιρετικό), το rw είναι υπεύθυνο για να βάζει την οθόνη σε read (VCC) ή σε write mode (Ground).

enable: Ο αριθμός του pin στον μικροελεγκτή που θα συνδεθεί το pin ενεργοποίησης (enable) της LCD οθόνης, το enable μας ενεργοποιεί την εγγραφή στην LCD οθόνη.

d0, d1, d2, d3, d4, d5, d6, d7: οι αριθμοί τον pin στον μικροελεγκτή που συνδέονται με τις αντίστοιχες ακίδες δεδομένων στην οθόνη LCD. Το d0, d1, d2 και d3 είναι προαιρετικά. αν

παραλειφθούν, η οθόνη LCD θα ελέγχεται χρησιμοποιώντας μόνο τις τέσσερις γραμμές δεδομένων (d4, d5, d6, d7).

π.χ. LiquidCrystal lcd(34,35,36,37,38,39);

Εφόσον εισάγουμε την βιβλιοθήκη της LCD και ορίσουμε τα κατάλληλα pins που θα χρησιμοποιηθούν, υπάρχουν κάποιες βασικές εντολές που ελέγχουν την LCD οθόνη που είναι οι παρακάτω:

Όνομα συνάρτησης / εντολής:

begin()

Περιγραφή συνάρτησης / εντολής:

Αρχικοποιεί τη διεπαφή στην οθόνη LCD και καθορίζει τις διαστάσεις (πλάτος και ύψος) της οθόνης. Η εντολή begin() πρέπει να καλείται πριν από οποιαδήποτε άλλη εντολή της βιβλιοθήκης LCD.

Σύνταξη εντολής:

lcd.begin(cols, rows)

Παράμετροι:

lcd: μια μεταβλητή τύπου LiquidCrystal

cols: ο αριθμός των στηλών που έχει η οθόνη

rows: ο αριθμός των γραμμών που έχει η οθόνη

Επιστρέφει:

Τίποτα

Παράδειγμα:

lcd.begin(16,2);

Αρχικοποιεί την διεπαφή της οθόνης, καθορίζοντας τις διαστάσεις της οθόνη που θα χρησιμοποιηθεί, ότι θα έχει 16 στήλες και 2 γραμμές.

Όνομα συνάρτησης / εντολής:

clear()

Περιγραφή συνάρτησης / εντολής:

Καθαρίζει την LCD οθόνη και τοποθετεί τον δρομέα/κέρσορα(cursor) στην επάνω αριστερή γωνία.

Σύνταξη εντολής:

lcd.clear()

Παράμετροι:

lcd: μια μεταβλητή τύπου LiquidCrystal

Επιστρέφει:

Τίποτα

Παράδειγμα:

lcd.clear();

Καθαρίζει την LCD οθόνη και τοποθετεί τον κέρσορα στην επάνω αριστερή γωνία.

Όνομα συνάρτησης / εντολής:

setCursor()

Περιγραφή συνάρτησης / εντολής:

Τοποθέτηση του κέρσορα (cursor) της LCD οθόνης. Δηλαδή, ορίζεται η νέα θέση στην οποία θα εμφανιστεί το επόμενο κείμενο που θα γραφτεί στην LCD οθόνη.

Σύνταξη εντολής:

lcd.setCursor(col, row)

Παράμετροι:

lcd: μια μεταβλητή τύπου LiquidCrystal

col: η στήλη στην οποία θα τοποθετηθεί ο cursor, με το 0 να είναι η πρώτη στήλη

row: η γραμμή στην οποία θα τοποθετηθεί ο cursor, με το 0 να είναι η πρώτη γραμμή

Επιστρέφει:

Τίποτα

Παράδειγμα:

lcd.setCursor(0,1);

Τοποθετεί τον cursor της LCD οθόνης στην πρώτη στήλη και δεύτερη γραμμή

Όνομα συνάρτησης / εντολής:

print()

Περιγραφή συνάρτησης / εντολής:

Γράφει/εκτυπώνει κείμενο στην LCD οθόνη

Σύνταξη εντολής:

lcd.print(data)

lcd.print(data, BASE)

Παράμετροι:

lcd: μια μεταβλητή τύπου LiquidCrystal

data: τα δεδομένα προς εκτύπωση (char, byte, int, long, ή string)

BASE (προαιρετικό): βάση για την εκτύπωση αριθμών:

- BIN για δυαδικό (βάση 2),
- DEC για δεκαδικό (βάση 10),
- OCT για οκταδικό (βάση 8),
- HEX για δεκαεξαδικό (βάση 16).

Επιστρέφει:

Byte

Η εντολή print () αν διαβαστεί, θα επιστρέψει τον αριθμό των bytes που έχουν γραφτεί, εάν και ο αριθμός αυτός είναι προαιρετικός.

Παράδειγμα:

lcd.print("Hello World");

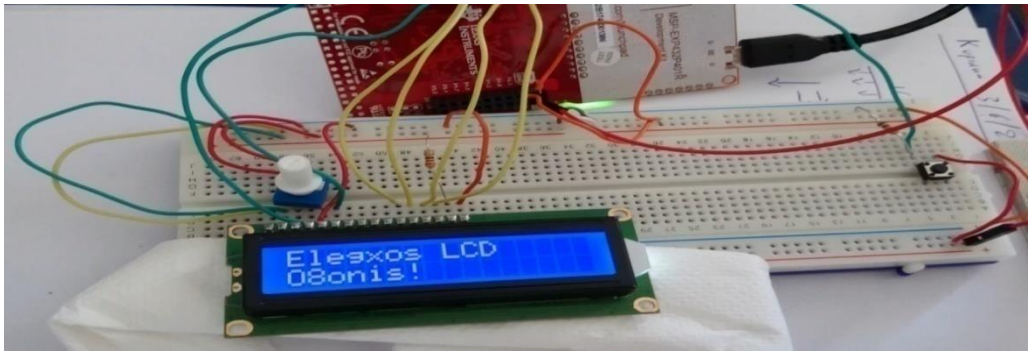
Μας εκτυπώνει στην LCD οθόνη το κείμενο Hello World.

Εφαρμογή LCD οθόνης

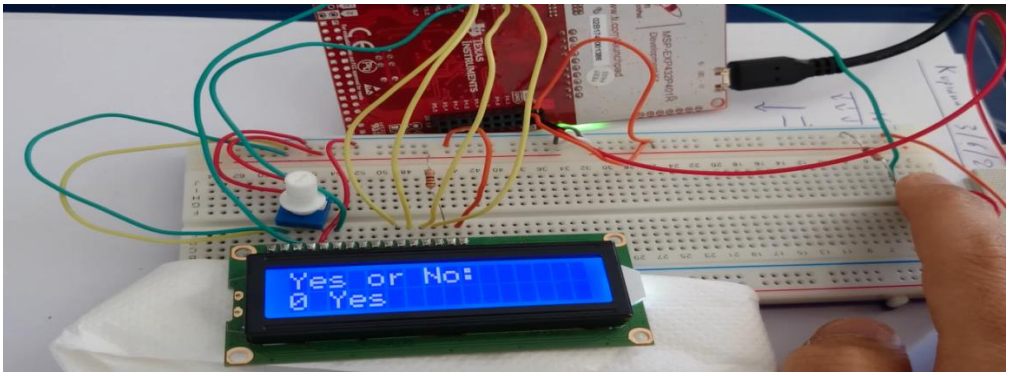
Ακολουθούν φωτογραφίες μιας εφαρμογής που μας κάνει τον έλεγχο μια LCD οθόνης, καθώς συνδυάζει όλες τις παραπάνω εντολές που είδαμε της βιβλιοθήκης LCD, τέλος γίνεται η υλοποίηση της σε Breadboard.

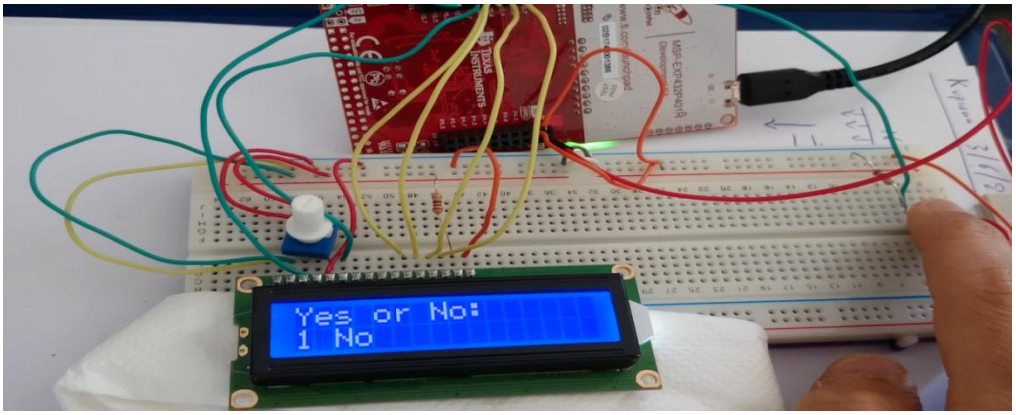
Η οθόνη αρχικοποιείται σε 4 bit mode χωρίς την γραμμή ελέγχου rw (το rw το βάλαμε στην γείωση δηλαδή write mode), ακολουθήσαμε την παρακάτω σύνταξη: LiquidCrystal όνομα μεταβλητής(rs, enable, d4, d5, d6, d7)

Η εφαρμογή όταν ξεκίνα εμφανίζει στην οθόνη το μήνυμα: Elegxos LCD O8onis! όπως φαίνεται στην παρακάτω φωτογραφία.



εφόσον πατηθεί ο διακόπτης που φαίνεται δεξιά της παραπάνω φωτογραφίας ενεργοποιείται η εντολή random() η οποία την έχουμε συντάξει ως εξής: int reply = random(2); , μας επιστρέφει τιμές από το 0 έως 1 στην ακέραια μεταβλητή reply, έπειτα μας εμφανίζει τα μηνύματα Yes αν είναι 0 και NO αν είναι 1. Ακολουθούν φωτογραφίες με τα μηνύματα.





Φωτογραφίες του κώδικα της εφαρμογής για να δείξουμε την σωστή σύνταξη των εντολών.

_11_LCD_Screen | Energia 1.6.10E18

File Edit Sketch Tools Help

_11_LCD_Screen §

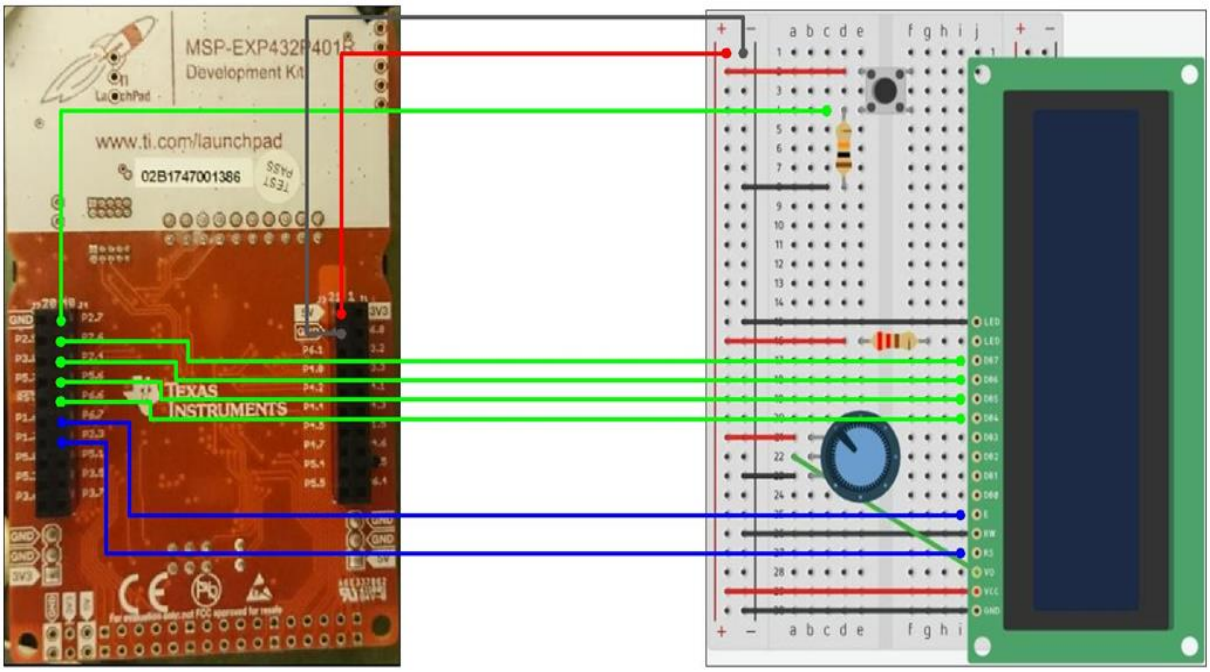
```
#include <LiquidCrystal.h>

//Sthn LCD function (rs,enable,d4,d5,d6,d7)

LiquidCrystal lcd(34,35,36,37,38,39);
const int switchPin = 40;
int switchState = 0;
int prevSwitchState = 0;
int reply;

void setup() {
  lcd.begin(16,2);
  pinMode(switchPin, INPUT);

  lcd.print("Elegxos LCD");
  lcd.setCursor(0,1);
  lcd.print("O8onis!");
}

void loop()
  switchState = digitalRead(switchPin);
  if(switchState != prevSwitchState) {
    if (switchState == LOW){
      reply = random(2);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Yes or No:");
```

_11_LCD_Screen | Energia 1.6.10E18

File Edit Sketch Tools Help

_11_LCD_Screen

```
11 void setup() {
12   lcd.begin(16,2);
13   pinMode(switchPin, INPUT);
14
15   lcd.print("Elegxos LCD");
16   lcd.setCursor(0,1);
17   lcd.print("O8onis!");
18 }
19
20 void loop() {
21   switchState = digitalRead(switchPin);
22   if(switchState != prevSwitchState) {
23     if (switchState == LOW){
24       reply = random(2);
25       lcd.clear();
26       lcd.setCursor(0,0);
27       lcd.print("Yes or No:");
28       lcd.setCursor(0,1);
29
30       switch(reply){
31         case 0: lcd.print(reply);lcd.print(" Yes"); break;
32         case 1: lcd.print("1 No"); break;
33       }
34     }
35   }
36   prevSwitchState = switchState;
37 }
```



Breadboard 3. Υλοποίηση εφαρμογής LCD οθόνης

Χρησιμοποιήθηκε 10 KΩ ποτενσιόμετρο για την ρύθμιση της αντίθεσης (contrast) της οθόνης που συνδέθηκε στο Vø pin της οθόνης, μια εξωτερική pulldown down αντίσταση 10 KΩ για το εξωτερικό κουμπί, καθώς και μια αντίσταση 220 Ω στο LED+ pin της οθόνης.

7ο Κεφάλαιο. Υλοποίηση εφαρμογής ταξινόμησης υλικών σε ράφια και κατασκευή της ραφιέρας

Η υλοποίηση εφαρμογής ταξινόμησης υλικών σε ράφια, χρειάστηκε κάποια επιπλέον εργαλεία εκτός του MSP-EXP432P401R όπως:

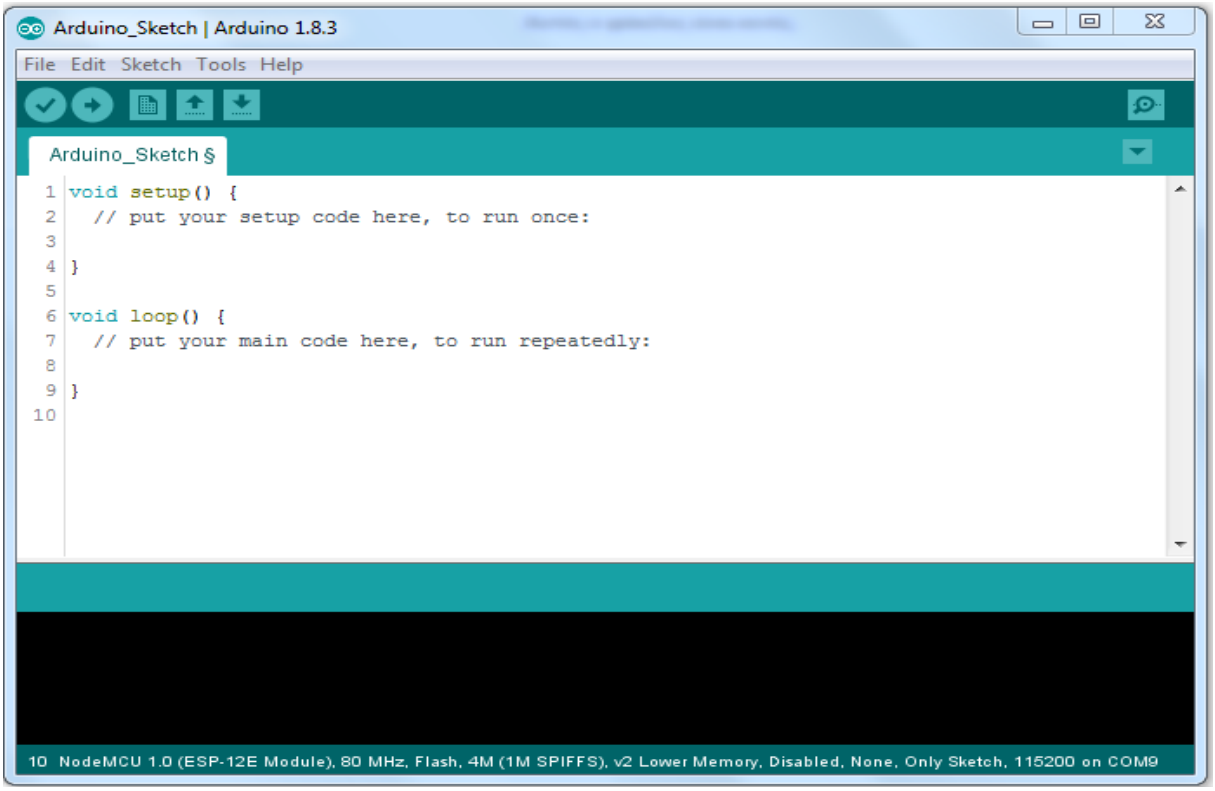
- Την εφαρμογή Blynk app στο κινητό τηλέφωνο.
- Έναν μικροελεγκτή με συνδεσιμότητα στο internet στην περίπτωση μας με χρήση wifi, δυνατότητα σύνδεσης του με την εφαρμογή Blynk, καθώς και σύνδεσή του με τον MSP-EXP432P401R, ένας τέτοιος μικροελεγκτής που χρησιμοποιήθηκε ήταν ο Arduino ESP8266.
- Μια χειροποίητη πλακέτα με αποκωδικοποιητές/αποπλέκτες για τον έλεγχο των RGB LED της ραφιέρας, για τον λόγο ότι δεν μας έφταναν τα I/O του MSP-EXP432P401R, καθώς μας δίνει και την δυνατότητα απομόνωσης των μικροελεγκτών από την υπόλοιπη κατασκευή για την δυνατότητα επαναπρογραμματισμού των μικροελεγκτών σε περίπτωση συντήρησης.
- Μια LCD οθόνη για της ένδειξης της ραφιέρας.
- Το πρόγραμμα PLX-DAQ-v2.11 που τρέχει στο Excel (Μακροεντολή) για να κάνει καταγραφή διάφορων τιμών της ραφιέρας, η σύνδεση γίνεται μέσω της σειριακής θύρας που συνδέεται ο μικροελεγκτής MSP-EXP432P401R με τον υπολογιστή.
- Τέλος έγινε η κατασκευή της ραφιέρας ώστε να μπορέσουμε να τρέξουμε την εφαρμογή πρακτικά.

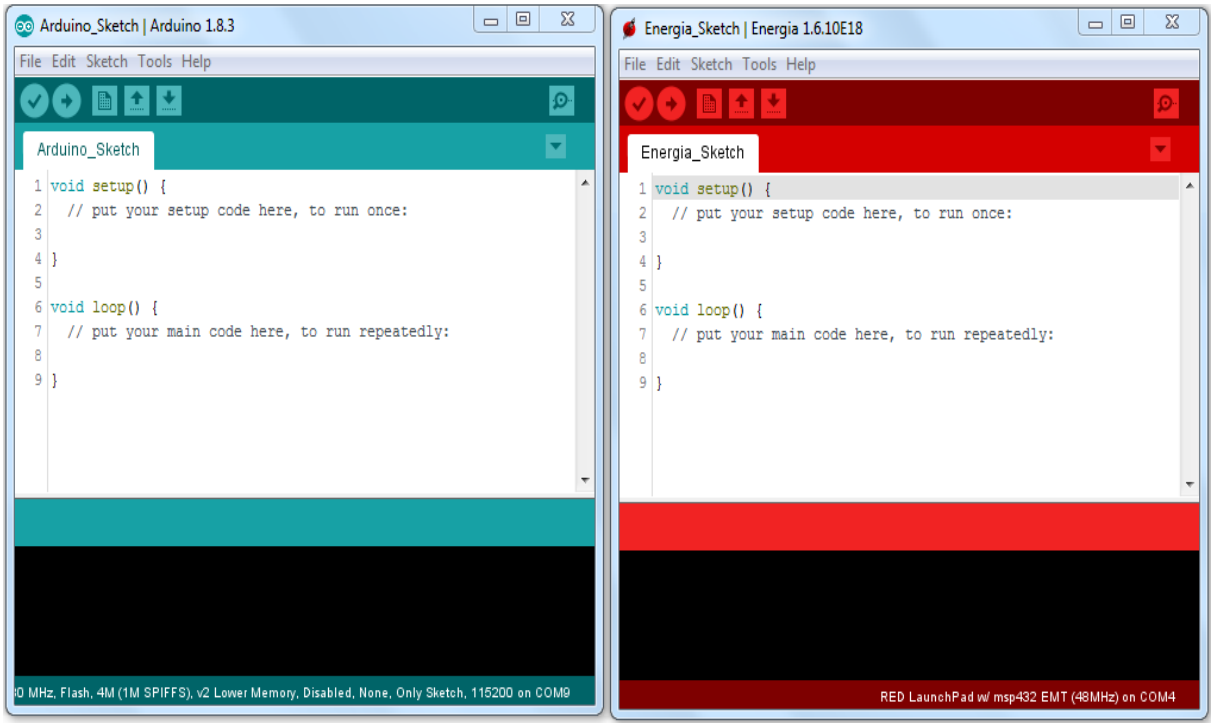
Θα αναλύσουμε παρακάτω της βασικές λειτουργίες που χρησιμοποιήθηκαν σε κάθε εργαλείο και στην συνέχεια θα γίνει η αναλυτική περιγραφή της λειτουργίας της ραφιέρας (εφαρμογή ταξινόμησης υλικών σε ράφια), επίσης θα γίνει και η περιγραφή κατασκευής της ραφιέρας.

Arduino ESP8266

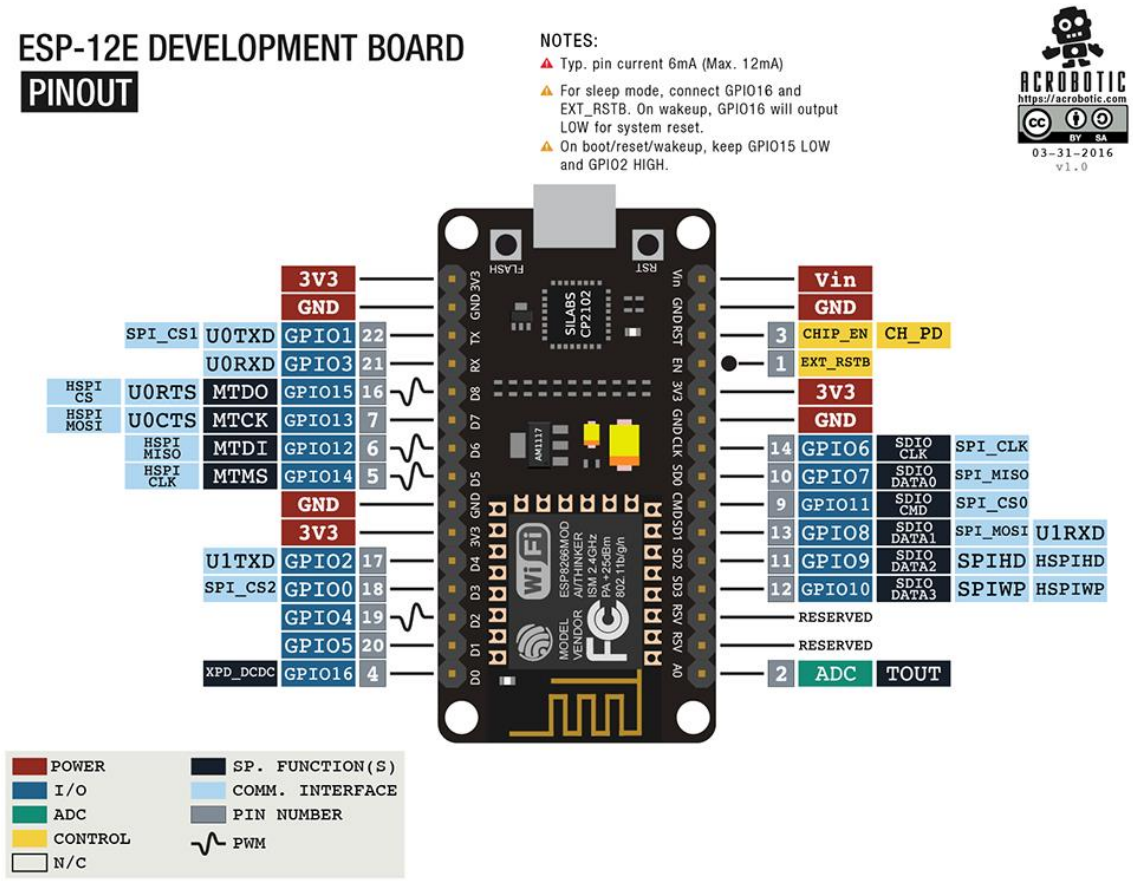
Για την διασύνδεση της ραφιέρας με το internet χρησιμοποιήθηκε ο Arduino ESP8266 και ποιό συγκεκριμένα το αναπτυξιακό Arduino ESP8266 NodeMCU 1.0 (ESP-12E Module), εκμεταλλεύοντας το ενσωματωμένο wifi του, καθώς ότι υποστηρίζεται από την εφαρμογή Blynk και ότι μπορεί να συνδεθεί εύκολα μέσω του πρωτόκολλου SPI με τον MSP-EXP432P401R.

Για τον προγραμματισμό του ESP8266 χρησιμοποιήθηκε το πολύ γνωστό ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) Arduino IDE, στο οποίο έχει βασιστεί και το Energia IDE το οποίο αναλύθηκε στο 6^{o} κεφάλαιο. Το Arduino IDE έχει ακριβώς της ίδιες εντολές/συναρτήσεις που είδαμε στο Energia IDE, καθώς και το ίδιο γραφικό περιβάλλον απλώς έχει άλλο χρώμα. Το Arduino IDE είναι open source. Ακολουθεί μια φωτογραφία του Arduino IDE και μια δεύτερη φωτογραφία που δείχνει της ομοιότητες του με το Energia IDE.

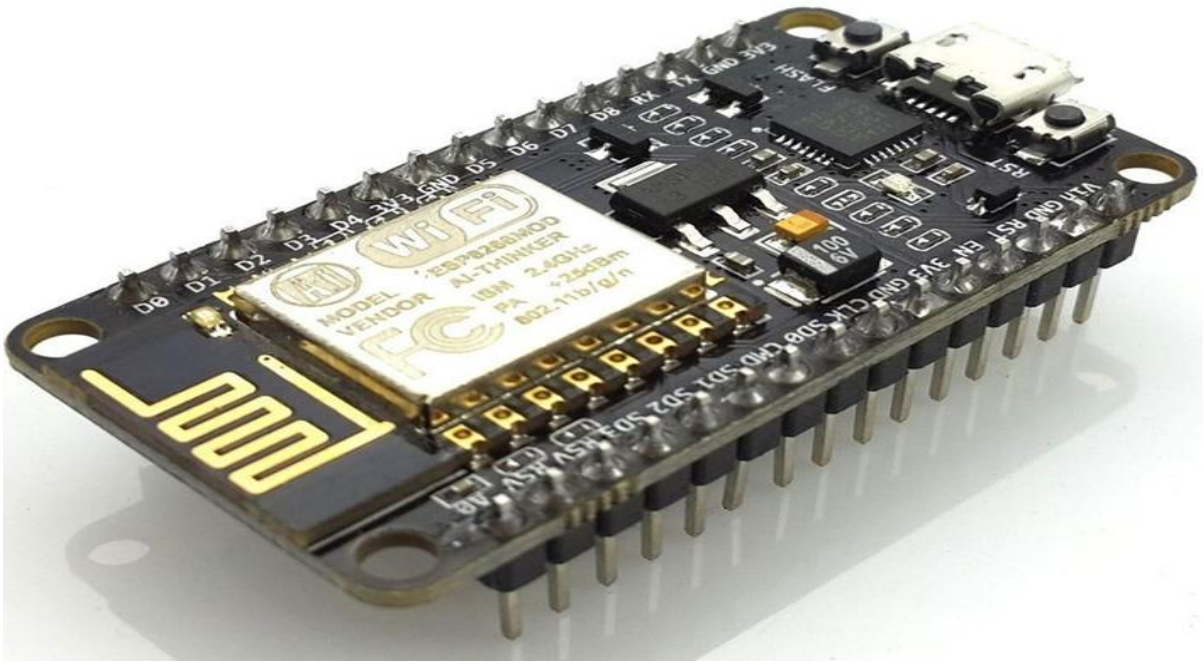




Ο χάρτης λειτουργιών τον ακροδεκτών (pins) του ESP8266 ESP-12E είναι:



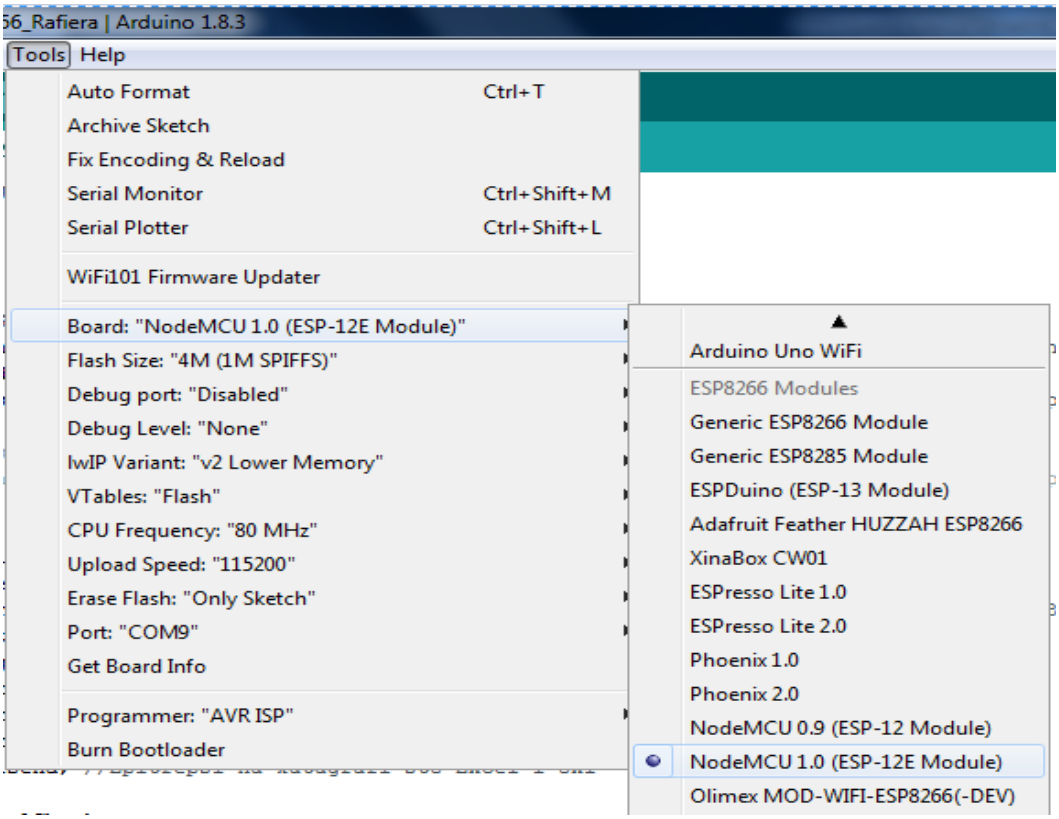
Εικόνα 57. Χάρτης λειτουργιών των Pin του ESP8266 ESP-12E



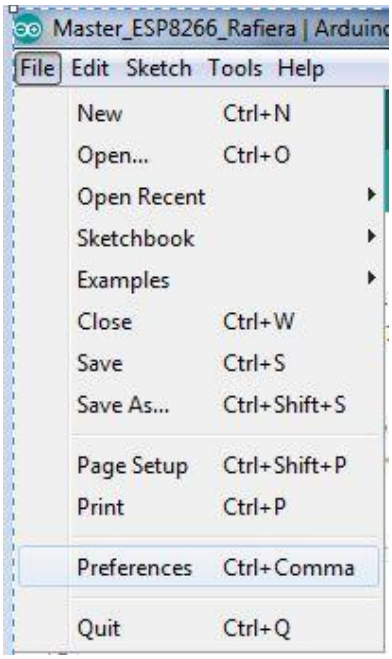
Εικόνα 58. Αναπτυξιακό ESP8266 ESP-12E

Ρυθμίζουμε τον Arduino IDE ώστε να αναγνωρίσει τον ESP8266, οι ρυθμίσεις γίνονται και αυτές ακριβώς όπως στον Energia IDE.

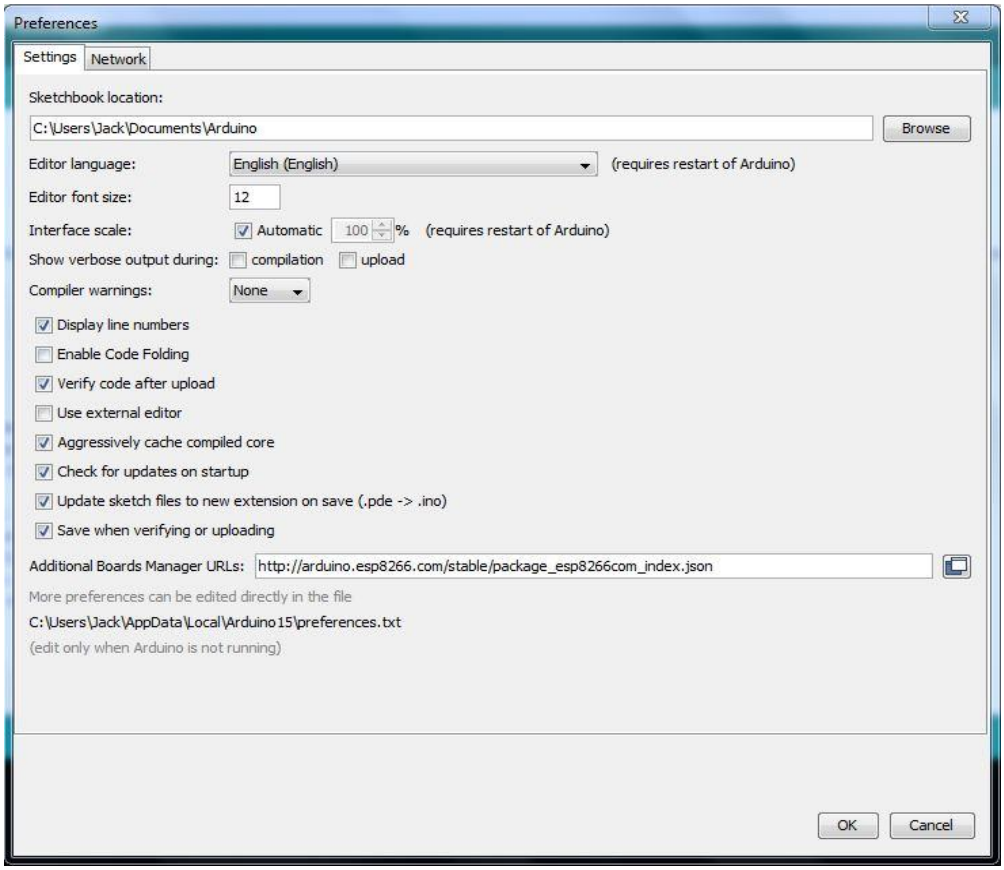
Πρώτα ρυθμίζουμε τον Board Manager, Tools>Board: "NodeMCU 1.0 (ESP-12E Module)":



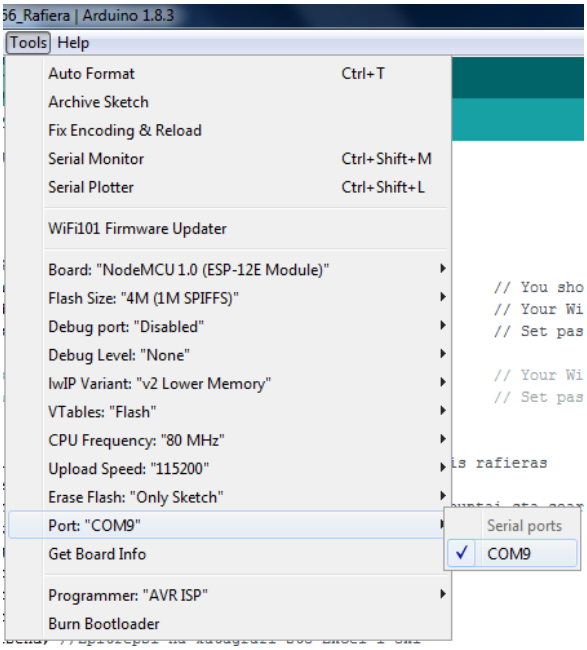
Αν δεν υπάρχει το Board "NodeMCU 1.0 (ESP-12E Module)", το εισάγουμε από το File>Preferences:



εισάγεται από εξωτερικό URL στην ρύθμιση με το όνομα: "Additional Boards Manager URLs: " στην δικιά μας περίπτωση χρησιμοποιήθηκε το URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json
όπως φαίνεται και στην παρακάτω εικόνα.



Στην συνέχεια ρυθμίζουμε την σειριακή θύρα, Tools>Port: "COMx" (οπού x ο αριθμός σειριακής θύρας):



Εφόσον γίνανε και οι βασικές ρυθμίσεις και αναγνώρισε ο Arduino IDE τον ESP8266, είναι έτοιμος για να τον προγραμματίσουμε.

Η σύνδεση του MSP-EXP432P401R με τον ESP8266 έγινε μέσω της σειριακής επικοινωνίας SPI, Ο Master είναι ο ESP8266 και ο Slave είναι ο MSP-EXP432P401R, από την μεριά του ESP8266 σαν Master, η επικοινωνία υλοποιήθηκε με την βοήθεια της βιβλιοθήκης SPI, από την μεριά του MSP-EXP432P401R σαν Slave, η επικοινωνία υλοποιήθηκε με την βοήθεια των καταχωρητών του SPI (η ανάλυση του SPI έγινε στο 5^{ο} κεφάλαιο).

Θα αναλύσουμε τις βασικές εντολές/συναρτήσεις της βιβλιοθήκης του SPI που χρησιμοποιήθηκαν.

Πρώτα γίνεται η εισαγωγή της βιβλιοθήκης κατά τον γνωστό τρόπο που είδαμε στο Energia IDE, #include <SPI.h> , στην συνέχεια βλέπουμε τις εντολές της.

Όνομα συνάρτησης / εντολής:

begin()

Περιγραφή συνάρτησης / εντολής:

Αρχικοποιεί το δίαυλο SPI ρυθμίζοντας το SCK, το MOSI και το SS σε εξόδους, τραβώντας το SCK και το MOSI σε Low και το SS HIGH. (Δηλαδή η αρχικοποίηση γίνεται σαν Master, γιατί παρατηρείτε ότι το MOSI είναι έξοδος δηλαδή Master OUT).

Σύνταξη εντολής:

SPI.begin()

Παράμετροι:

Κανένας

Επιστρέφει:

Τίποτα

Παράδειγμα:

SPI.begin();

Γίνεται η αρχικοποίηση του διαύλου του SPI (σαν Master)

Όνομα συνάρτησης / εντολής:

setBitOrder()

Περιγραφή συνάρτησης / εντολής:

Ορίζει τη σειρά των bits που μετατοπίζονται από και προς το δίαυλο SPI, LSBFIRST (ελάχιστο σημαντικό ψηφίο πρώτο) ή MSBFIRST (το πιο σημαντικό ψηφίο πρώτο).

Σύνταξη εντολής:

SPI.setBitOrder(order)

Παράμετροι:

order: LSBFIRST ή MSBFIRST

Επιστρέφει:

Τίποτα

Παράδειγμα:

SPI.setBitOrder(MSBFIRST);

Ορίζει τη σειρά των bits που μετατοπίζονται από και προς το δίαυλο SPI σαν MSBFIRST (το πιο σημαντικό ψηφίο πρώτο).

Όνομα συνάρτησης / εντολής:

setDataMode()

Περιγραφή συνάρτησης / εντολής:

Ορίζει τη λειτουργία δεδομένων SPI: δηλαδή την πολικότητα του ρολογιού (clock polarity) και τη φάση του (clock phase).

Σύνταξη εντολής:

SPI.setDataMode(mode)

Παράμετροι:

mode: SPI_MODE0, SPI_MODE1, SPI_MODE2, SPI_MODE3 (αναλυτικότερα Εικόνα 52. διάγραμμα χρονισμού του SPI).

Επιστρέφει:

Τίποτα

Παράδειγμα:

SPI.setDataMode(SPI_MODE0);

Ορίζει τη λειτουργία δεδομένων SPI σε MODE0 που αυτό σημαίνει ότι:

- Clock Phase: Τα δεδομένα αλλάζουν στην πρώτη άκρη του παλμού του ρολογιού και καταγράφονται στην επόμενη άκρη.
- Clock Polarity: Η ανενεργή κατάσταση είναι η low.

Όνομα συνάρτησης / εντολής:

transfer()

Περιγραφή συνάρτησης / εντολής:

Η μεταφορά SPI βασίζεται στην ταυτόχρονη αποστολή και λήψη: τα λαμβανόμενα δεδομένα επιστρέφονται σε μια μεταβλητή (στέλνετε ή λαμβάνετε στον δίαυλο μέχρι 1 byte)

Σύνταξη εντολής:

SPI.transfer(val)

receivedVal = SPI.transfer(val)

Παράμετροι:

val: Το byte το που στέλνετε μέσω του διαύλου

Επιστρέφει:

Τα ληφθέντα δεδομένα

Παράδειγμα:

SPI.transfer(50);

Στέλνουμε τον αριθμό 50 σε μορφή byte (8bit), δηλαδή στο δυαδικό είναι: 0b00110010

Εφαρμογή Blynk

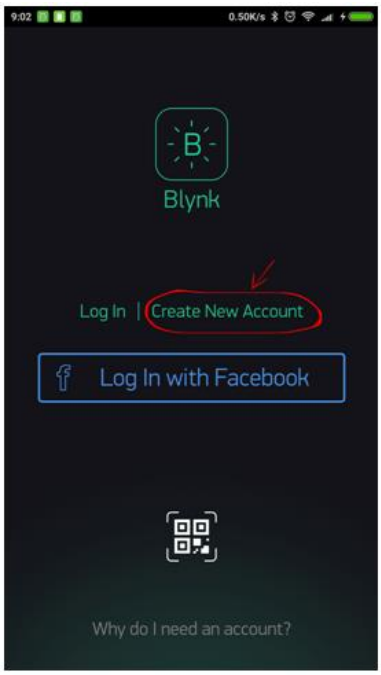
Η εφαρμογή Blynk είναι μια εφαρμογή κινητού τηλεφώνου που μας δίνει την δυνατότητα να ελέγχουμε συμβατούς μικροελεγκτές εξ αποστάσεως. Συγκεκριμένα λειτουργεί με συσκευές που υποστηρίζουν τουλάχιστον ένα από τα παρακάτω, Ethernet, WiFi, GPRS, Bluetooth ή και USB, στην δικιά μας περίπτωση χρησιμοποιήθηκε το WiFi.

Η εφαρμογή σου δίνει κάποιους πόντους δωρεάν που με αυτούς αγοράζεις τα εργαλεία (Widgets) που θα χρησιμοποιήσεις, αλλά αν θες περισσότερους τους αγοράζεις με χρήματα, σου δίνει δικαίωμα επιστροφής εργαλείου και σου επιστρέφει τους πόντους του.

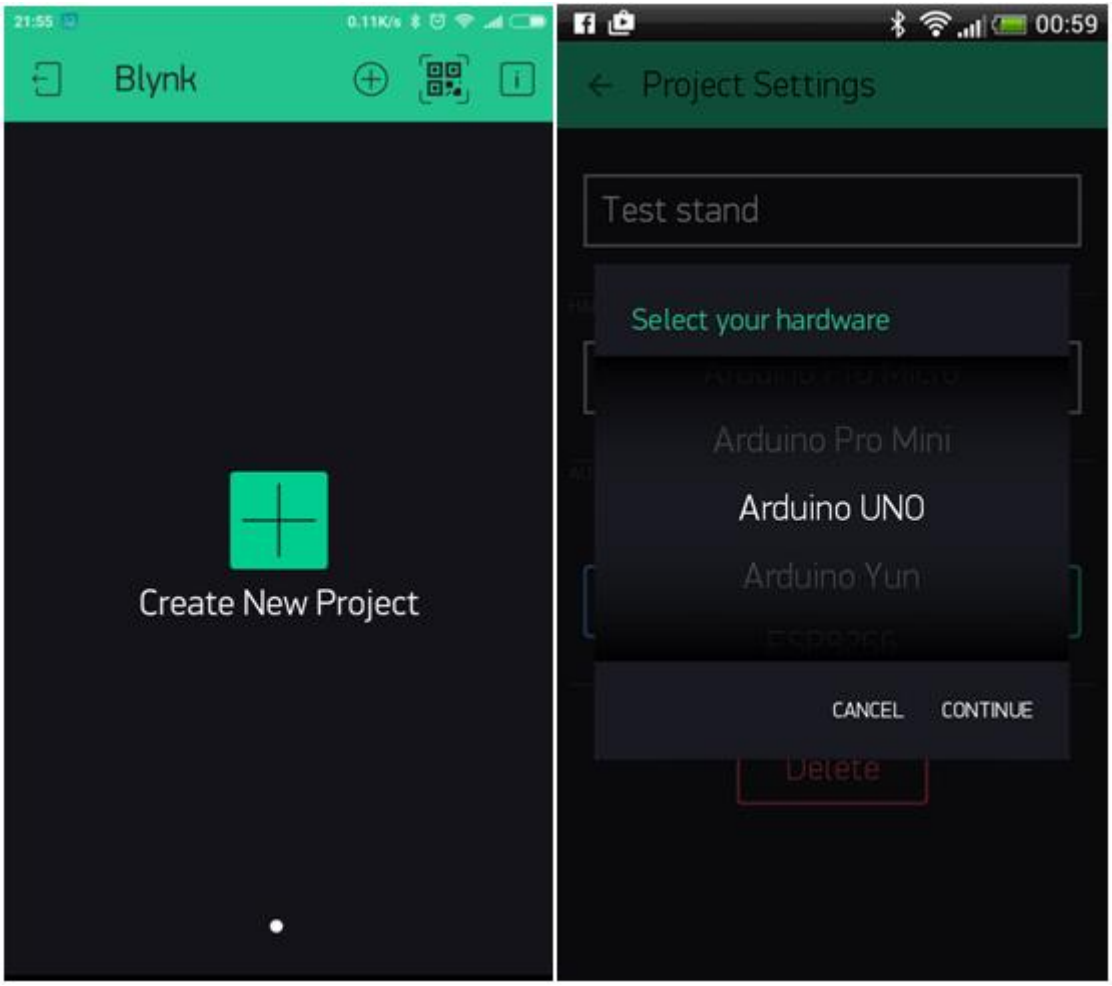
Στην αρχή κατεβάζουμε την εφαρμογή για κινητά Android ή IOS (Iphone) μέσω του Google play ή του App Store αντίστοιχα.

Για να συνδέσουμε το Blynk App και τον μικροελεγκτή μας, χρειαζόμαστε ένα Auth Token (άδεια χρήσης), το οποίο για να το πάρουμε κάνουμε τα εξής βήματα:

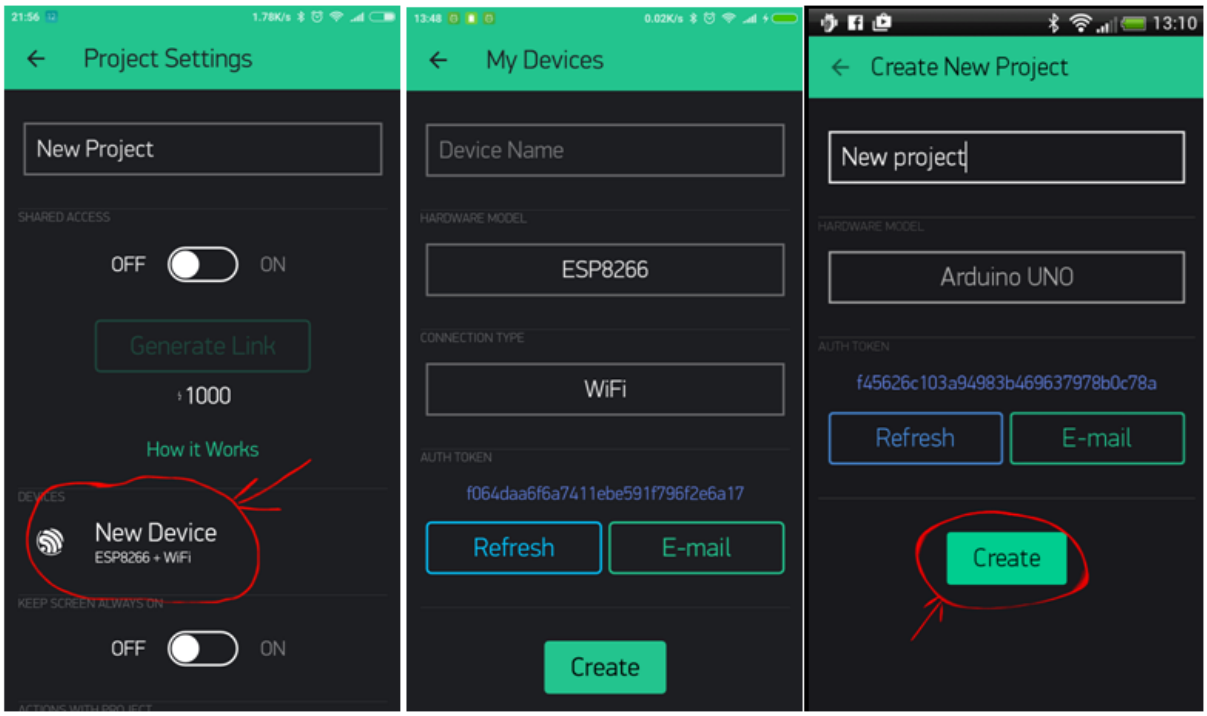
- Δημιουργούμε έναν νέο λογαριασμό στο Blynk App



- Στην συνέχεια δημιουργούμε ένα νέο πρόγραμμα (project) στην εφαρμογή, έπειτα επιλέγουμε την πλακέτα (board) που θα χρησιμοποιήσουμε.



- Μετά την δημιουργία του project, θα σταλθεί το Auth Token στο email που δηλώσαμε όταν δημιουργήσαμε τον λογαριασμό.



- Ελέγχουμε το Email μας και βρίσκουμε το Auth Token που θα χρησιμοποιήσουμε.

Στην συνέχεια κατεβάζουμε την βιβλιοθήκη του Blynk library από το επίσημο site: https://www.blynk.cc/getting-started/ (έχει αναλυτικά όλα τα βήματα που περιγράφουμε), εφόσον την κατεβάσουμε, την εγκαθιστούμε στον Arduino IDE ακριβώς με την ίδια διαδικασία που αναλύθηκε στο Energia IDE, βάζοντας την πρώτα στον φάκελο με το όνομα libraries ο οποίος βρίσκεται στους φακέλους εγκατάστασης του Arduino IDE.

Εφόσον εγκαταστήσουμε και την βιβλιοθήκη, έχουμε και το Auth Token στο email μας είμαστε σε θέση να αρχίσουμε να γράφουμε το πρόγραμμα στον μικροελεγκτής μας εκμεταλλεύοντας της επιπλέον εντολές που μας προσφέρει το Blynk, π.χ. να δημιουργήσουμε ένα κουμπί ή ότι άλλο εργαλείο (Widget) μας προσφέρει το Blynk στο κινητό μας και με την σύνδεση του κινητού και του μικροελεγκτή στο Internet να δημιουργούμε εφαρμογές που θα ελέγχουν αντικείμενα εξ αποστάσεως, αν θέλουμε περνούμε και διάφορες ενδείξεις στο κινητό π.χ. εικονικό LED στο κινητό μας να μας δείχνει ότι κάτι λειτουργεί ή ότι είναι σβηστό, αυτό ονομάζεται διαδίκτυο των αντικειμένων [Internet of Things(IoT)].

Διαλέξαμε το Board μας (ESP8266) και τη σύνδεση θα χρησιμοποιήσουμε (WiFi, Ethernet, Bluetooth κ.α), στην περίπτωση μας WiFi όπως προαναφέραμε.

Η βασική αρχικοποίηση που γίνεται στον ESP8266 είναι να εισάγουμε της δύο βασικές βιβλιοθήκες για να λειτουργήσει το Blynk, την βιβλιοθήκη του Blynk και την βιβλιοθήκη του WiFi.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

Στην συνέχεια αρχικοποιούμε της βιβλιοθήκες:

```
char auth[] = "YourAuthToken";
```

Αντιγράφουμε το Auth Token που μας στάλθηκε στο email μας.

```
char ssid[] = "Όνομα δικτύου";
```

Δίνουμε το όνομα του ασύρματου δικτύου προς χρήση

```
char pass[] = "Κωδικός δικτύου";
```

Τον κωδικό αυτού του δικτύου, αν δεν υπάρχει κωδικός, δηλαδή είναι ελεύθερο δίκτυο συντάσσετε έτσι: char pass[] = "";

Έπειτα χρησιμοποιήθηκε η εντολή αρχικοποίησης του Blynk μέσα στην setup() με το όνομα Blynk.begin(), δίνοντας της παραμέτρους που αρχικοποιηθήκαν παραπάνω:

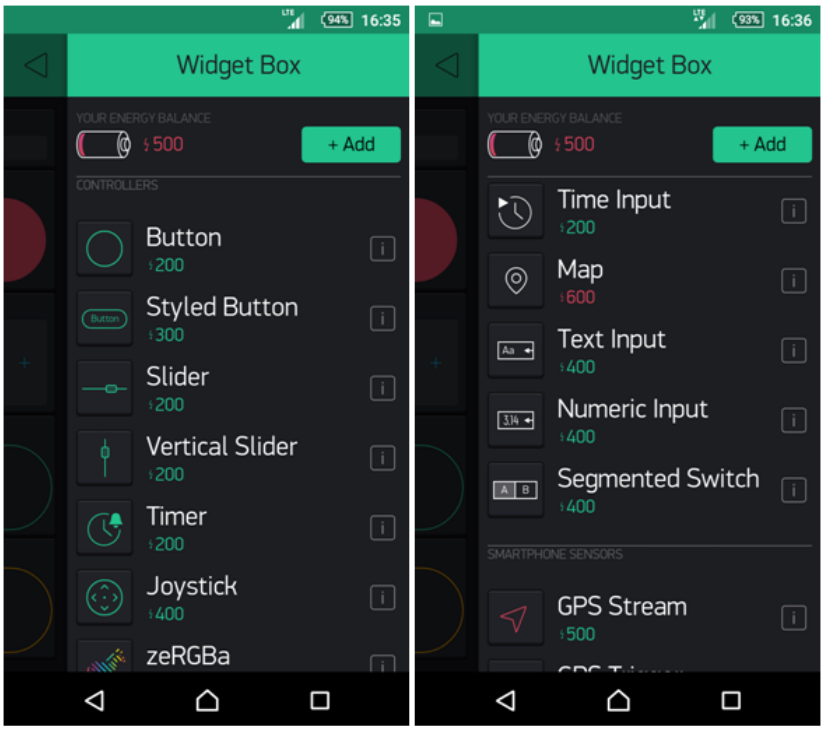
```
void setup() {
Blynk.begin(auth, ssid, pass);
}
```

Τέλος για να γίνει εκκίνηση του Blynk, γράφουμε μέσα στην loop() την εντολή Blynk.run():

```
void loop(){
Blynk.run();
}
```

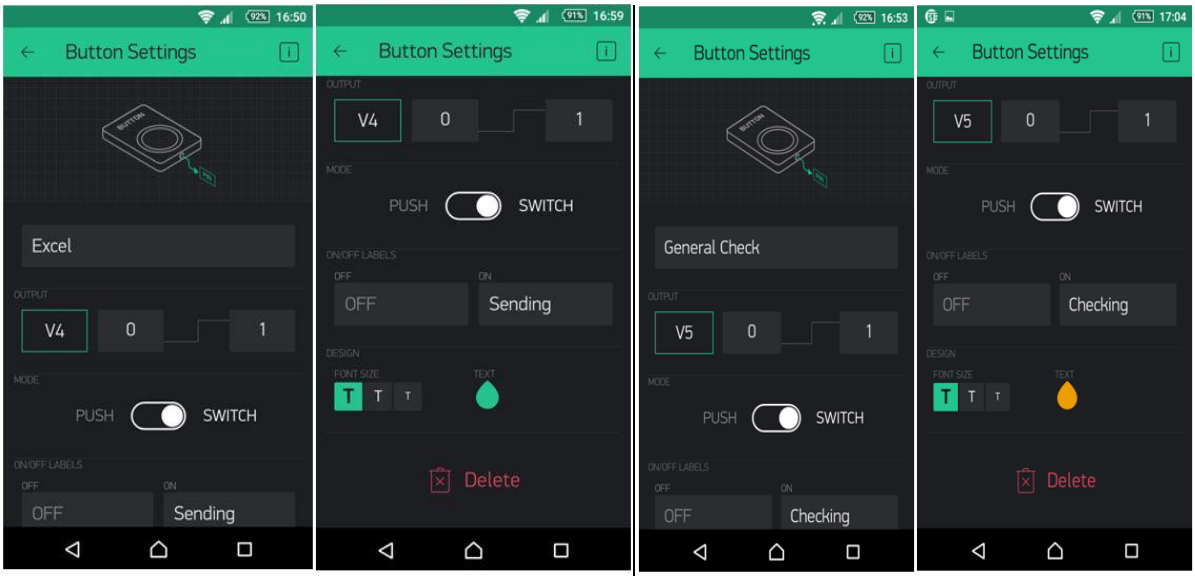
Στο κινητό μας τρέχουμε το Blynk και θα δούμε τα βασικά εργαλεία (Widgets) που χρησιμοποιήθηκαν όπως: το απλό Button, το Styled Button, το Text Input, το Numeric Input, στο επίσημο site του Blynk έχει αναλυτικά την λειτουργία όλων των Widget.

Ακολουθούν φωτογραφίες με το Widget Box που περιέχει τα εργαλεία (Widget) του Blynk, τα εργαλεία απλά πατάμε πάνω τους και τα σέρνουμε μέσα στο καμβά του προγράμματος που φτιάχνουμε (Drag and Drop).



Το κάθε Widget έχει τις δικιές του ρυθμίσεις (settings), θα δούμε τις βασικές ρυθμίσεις για τα Widget που χρησιμοποιήθηκαν στην εφαρμογή της ραφιέρας.

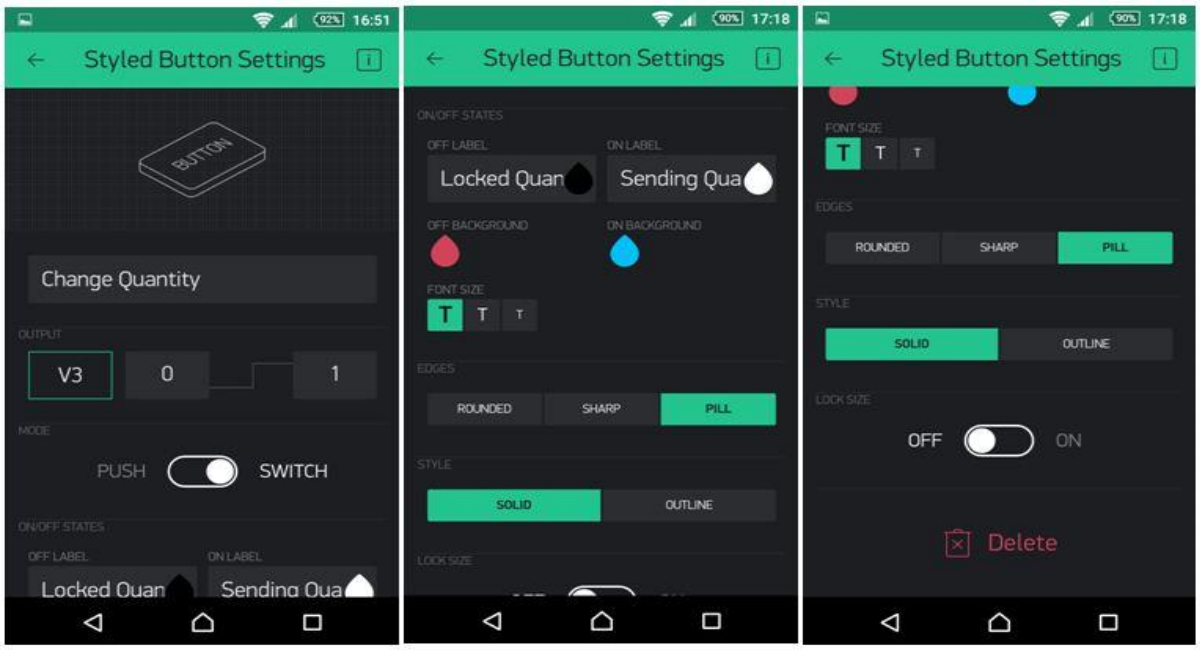
Τα απλά **Button Widget** ρυθμιστήκαν ως εξής:



Τους δόθηκαν οι ονομασίες Excel και General Check, στην συνέχεια τους δώσαμε την τιμή του διακόπτη όταν είναι OFF να είναι 0 και όταν είναι ON να είναι 1, αυτές οι τιμές αποθηκεύονται στης εικονικές (Virtual) μεταβλητές με τις ονομασίες V4 και V5, διαλέξαμε την λειτουργία διακόπτη (SWITCH) και στα δύο Button, έπειτα διαλέξαμε ετικέτα (Label)

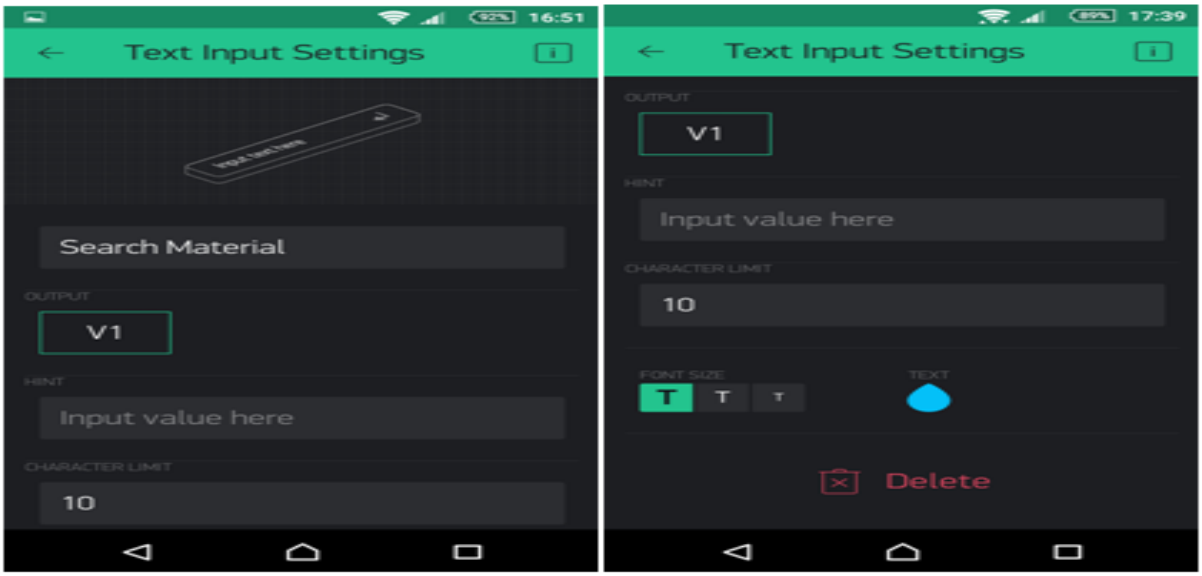
που θα φαίνεται όταν οι διακόπτες είναι ON ή OFF, τέλος έγινε η επιλογή μεγέθους και χρώματος γραμμάτων.

Το **Styled Button Widget** ρυθμίστηκε ως εξής:



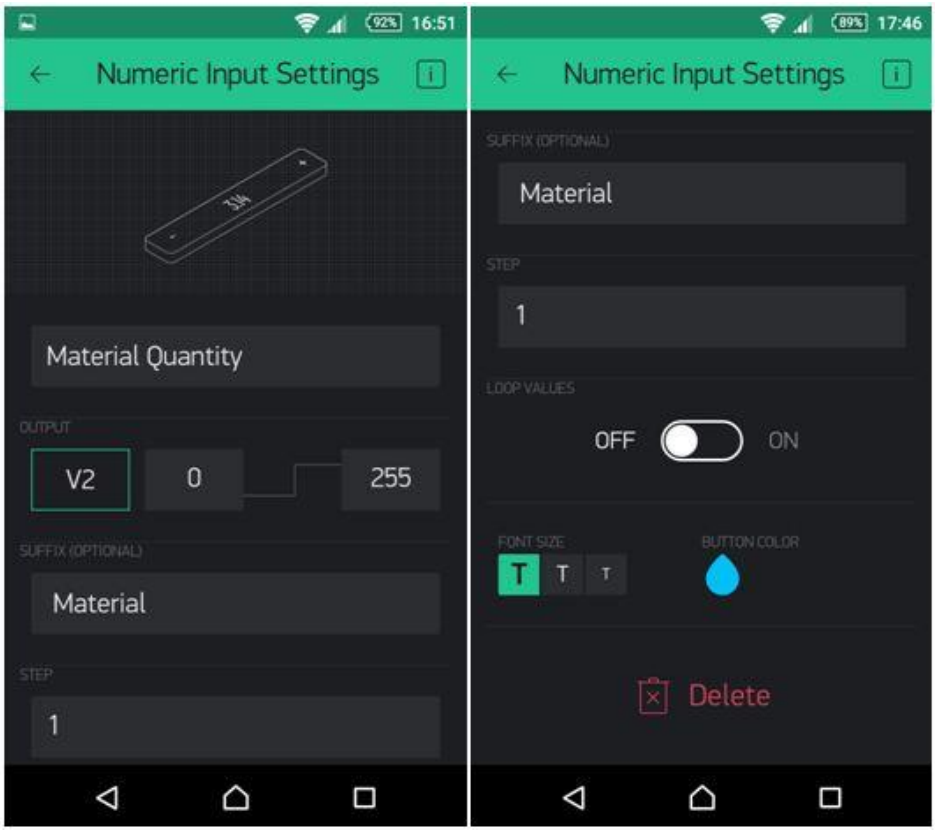
Του δόθηκε η ονομασία Change Quantity, αποθηκεύτηκε στην εικονική μεταβλητή V3, τιμή OFF = 0, τιμή ON = 1, είναι διακόπτης (SWITCH), έπειτα διαλέξαμε ετικέτα που θα φαίνεται όταν ο διακόπτης είναι ON ή OFF, έγινε η ρύθμιση μεγέθους και χρώματος γραμμάτων, μέχρι και εδώ είναι σαν απλό Button Widget, αλλά το Styled Button Widget έχει παραπάνω ρυθμίσεις όπως: έχει αλλαγή χρώματος φόντου (Background) διαφορετικό σε ON ή σε OFF, ρύθμιση σχήματος γωνιών του Widget π.χ. FILL και ρύθμιση του στυλ π.χ. αν είναι συμπαγές (SOLID).

Το **Text Input Widget** ρυθμίστηκε ως εξής:



Του δόθηκε η ονομασία Search Material, αποθηκεύτηκε στην εικονική μεταβλητή V1, ρυθμίστηκε το όριο χαρακτήρων που μπορούμε να γράψουμε στους 10 χαρακτήρες, ρυθμίστηκε μέγεθος και χρώμα γραμμάτων.

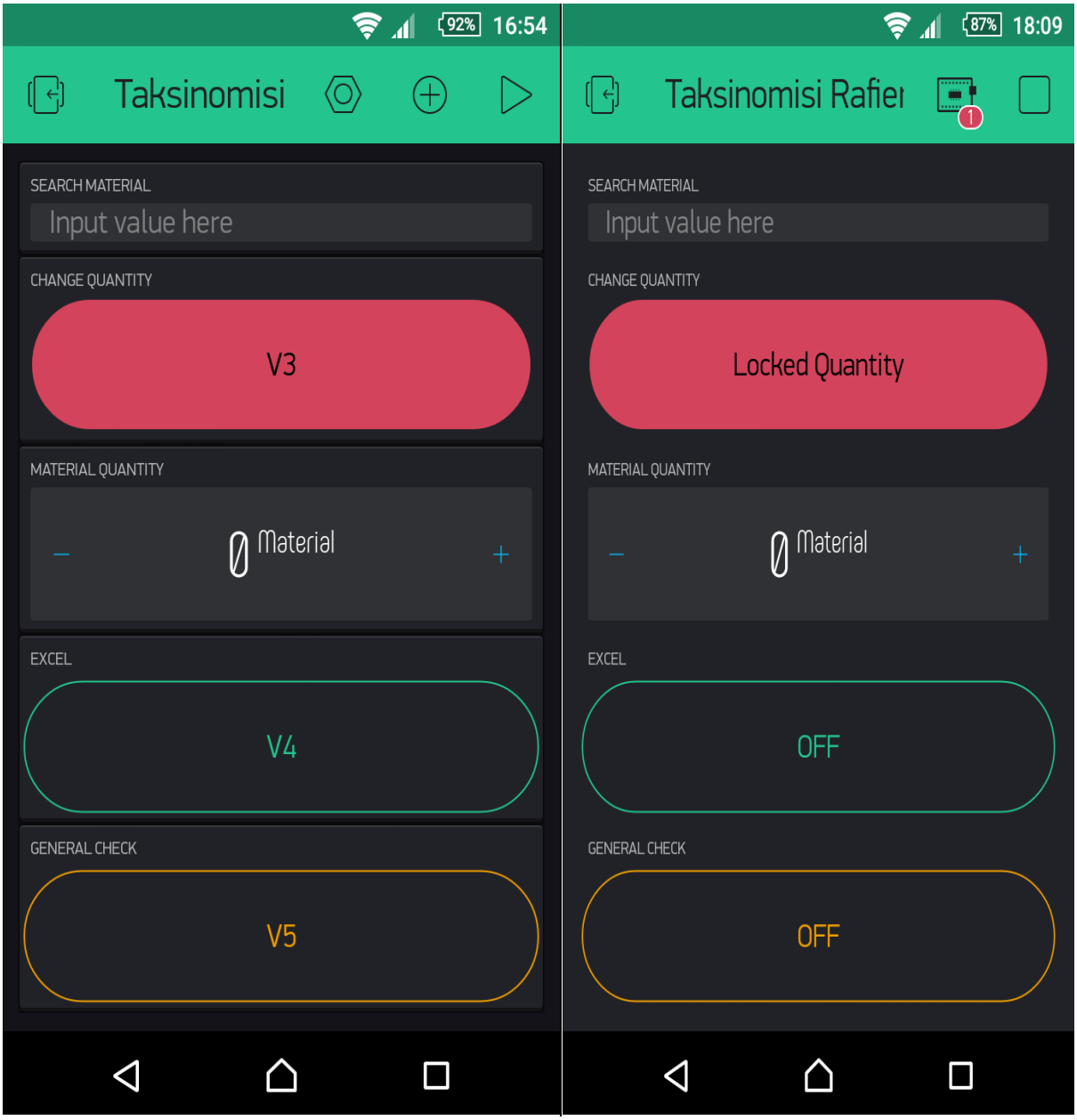
Το **Numeric Input Widget** ρυθμίστηκε ως εξής:



Του δόθηκε η ονομασία Material Quantity, αποθηκεύτηκε στην εικονική μεταβλητή V2, ρυθμίστηκε το όριο τιμών που μπορούμε να γράψουμε από τον αριθμό 0 έως τον 255, την κατάληξη (SUFFIX) του αριθμού την ονομάσαμε Material (είναι προαιρετικό), το βήμα (STEP) το ρυθμίσαμε στο 1, τιμές βρόχου (Loop Values) στο OFF, τέλος ρυθμίστηκε μέγεθος και χρώμα γραμμάτων.

Για όλα τα Widgets υπάρχει πάνω δεξιά στις ρυθμίσεις το σύμβολο  που είναι η βοήθεια (help) του συγκεκριμένου Widget για το πως λειτουργεί.

Άρα χρησιμοποιώντας αυτά τα Widget υλοποιήθηκε η διεπαφή της εφαρμογής μας στο κινητό τηλέφωνο. Ακολουθεί φωτογραφία του γραφικού περιβάλλοντος της εφαρμογής: Ταξινόμηση υλικών σε ράφια. Στο Blynk ονομάστηκε: Taksinomisi Rafieras. Στην αριστερή μεριά της φωτογραφίας, η εφαρμογή είναι σταματημένη και μπορούμε να την επεξεργαστούμε να της κάνουμε αλλαγές, ενώ στην δεξιά μεριά είναι εν ώρα λειτουργίας.



Εικόνα 59. Γραφικό περιβάλλον εφαρμογής: Ταξινόμηση Υλικών σε Ράφια

Στην αριστερή μεριά της Εικόνας 59. πατώντας το κουμπί του play  αρχίζει να λειτουργεί η εφαρμογή βλέπε δεξιά μεριά της Εικόνας 59. και ότι τιμές βάζουμε στις μεταβλητές στέλνονται στον Server του Blynk που ανήκει στους κατασκευαστές του Blynk και στην συνέχεια από τον Server στέλνονται στον μικροελεγκτή μας, αν είναι συνδεμένος σωστά με το Blynk όπως είδαμε στην αρχικοποίηση με το Auth token, το εικονίδιο  μας δείχνει ότι ο ESP8266 δεν είναι συνδεμένος αυτή την στιγμή στο Blynk.

Όλες οι Virtual μεταβλητές από τα Widget μεταφέρονται μέσω του Server στον μικροελεγκτή μας (ESP8266) και μέσω των εντολών τις βιβλιοθήκης του Blynk τις εκμεταλλευόμαστε για να φτιάξουμε ότι εφαρμογή θέλουμε, στην δικιά μας περίπτωση χρησιμοποιήθηκαν για να υλοποιηθεί η εφαρμογή Ταξινόμηση Υλικών σε Ράφια που θα αναλυθεί η λειτουργία της σε επόμενο υποκεφάλαιο του 7^{ου} κεφαλαίου.

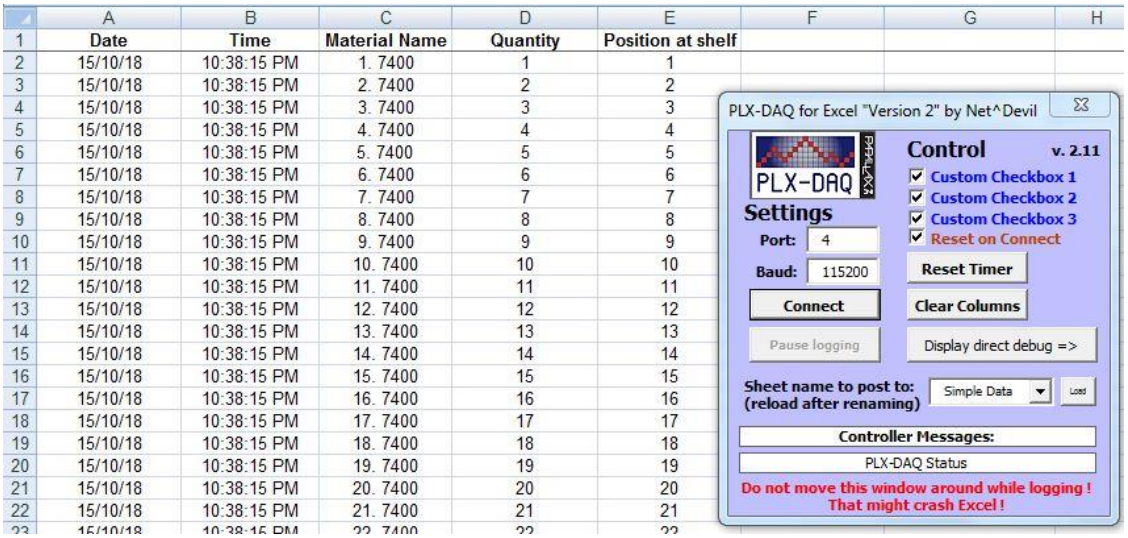
Η εντολή της βιβλιοθήκης Blynk που εκμεταλλεύεται τις Virtual μεταβλητές από το Blynk είναι η BLYNK_WRITE(Όνομα Virtual Μεταβλητής που δηλώθηκε στο κινητό), ακολουθεί παράδειγμα πως γίνεται η δόμηση της και ποιά είναι η θέση της μέσα στον κώδικα [μετά την setup(), πριν την loop()]:

```
String searchName; ←—— δηλώνουμε την String μεταβλητή που θα αποθηκευτεί το
                       αποτέλεσμα από το Blynk.
int quantity;  ←—————— δηλώνουμε την int μεταβλητή που θα αποθηκευτεί το
                       αποτέλεσμα από το Blynk.
void setup() {
//Διάφορες εντολές αρχικοποίησης.
}
//Από το Blynk Widget Text Input (λόγω ότι βάλαμε την V1).
BLYNK_WRITE(V1)
{
 searchName = param.asStr(); ←—— με το param.asStr(); παίρνουμε το εισερχόμενο String
                                 από την Virtual μεταβλητή V1 και το αποθηκεύουμε
                                 στην String μεταβλητή searchName.
}
//Από το Blynk Widget Numeric Input (λόγω ότι βάλαμε την V2).
BLYNK_WRITE(V2)
{
 quantity = param.asInt(); ←—— με το param.asInt(); παίρνουμε το εισερχόμενο int
                               από την Virtual μεταβλητή V2 και το αποθηκεύουμε
                               στην int μεταβλητή quantity.
}
void loop(){
//Διάφορες εντολές του προγράμματος
}
```

Καταγραφή στο Microsoft Excel με το PLX-DAQ-v2.11

Το PLX DAQ v2 είναι ένα πρόγραμμα το οποίο χρησιμοποιείται για την εγκαθίδρυση εύκολης επικοινωνίας μεταξύ του Microsoft Excel σε έναν υπολογιστή με λογισμικό Windows, με οποιαδήποτε συσκευή που υποστηρίζει πρωτόκολλο σειριακής θύρας.

Εμείς θα το χρησιμοποιήσουμε το PLX DAQ v2 για να επιτρέψουμε την επικοινωνία μεταξύ του MSP-EXP432P401R και του Excel, για να κάνουμε καταγραφή των μετρήσεων της ραφιέρα.



	A	B	C	D	E
1	Date	Time	Material Name	Quantity	Position at shelf
2	15/10/18	10:38:15 PM	1. 7400	1	1
3	15/10/18	10:38:15 PM	2. 7400	2	2
4	15/10/18	10:38:15 PM	3. 7400	3	3
5	15/10/18	10:38:15 PM	4. 7400	4	4
6	15/10/18	10:38:15 PM	5. 7400	5	5
7	15/10/18	10:38:15 PM	6. 7400	6	6
8	15/10/18	10:38:15 PM	7. 7400	7	7
9	15/10/18	10:38:15 PM	8. 7400	8	8
10	15/10/18	10:38:15 PM	9. 7400	9	9
11	15/10/18	10:38:15 PM	10. 7400	10	10
12	15/10/18	10:38:15 PM	11. 7400	11	11
13	15/10/18	10:38:15 PM	12. 7400	12	12
14	15/10/18	10:38:15 PM	13. 7400	13	13
15	15/10/18	10:38:15 PM	14. 7400	14	14
16	15/10/18	10:38:15 PM	15. 7400	15	15
17	15/10/18	10:38:15 PM	16. 7400	16	16
18	15/10/18	10:38:15 PM	17. 7400	17	17
19	15/10/18	10:38:15 PM	18. 7400	18	18
20	15/10/18	10:38:15 PM	19. 7400	19	19
21	15/10/18	10:38:15 PM	20. 7400	20	20
22	15/10/18	10:38:15 PM	21. 7400	21	21

Εικόνα 60. Μετρήσεις Ραφιέρας στο PLX-DAQ-v2.11

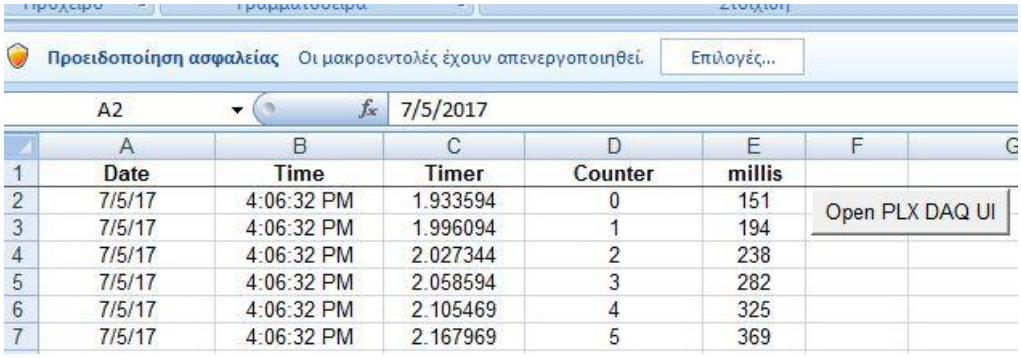
Όλη η επικοινωνία γίνεται με τις εντολές Serial.print και Serial.println, όπως ακριβώς χρησιμοποιούνται από τον MSP-EXP432P401R στο Energia IDE για παρακολούθηση στο Serial Monitor.

Θα δούμε το πως λειτουργεί το PLX DAQ v2, θα δούμε στις εντολές Serial.print και Serial.println το τη γράφουμε για να πάρουμε τις μετρήσεις της ραφιέρας στο Excel. Θα αναλυθεί μονό ότι χρησιμοποιήθηκε για την ραφιέρα, πλήρης οδηγός υπάρχει στο συνοδευτικό αρχείο Word, με το που κατεβάσουμε (Download) το πρόγραμμα PLX DAQ v2, είναι open source πρόγραμμα.

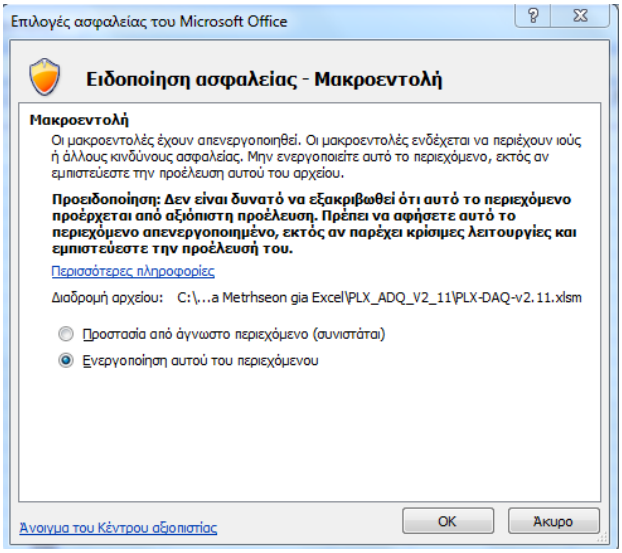
Πως λειτουργεί:

Πρώτον ανοίγουμε το αρχείο του PLX 

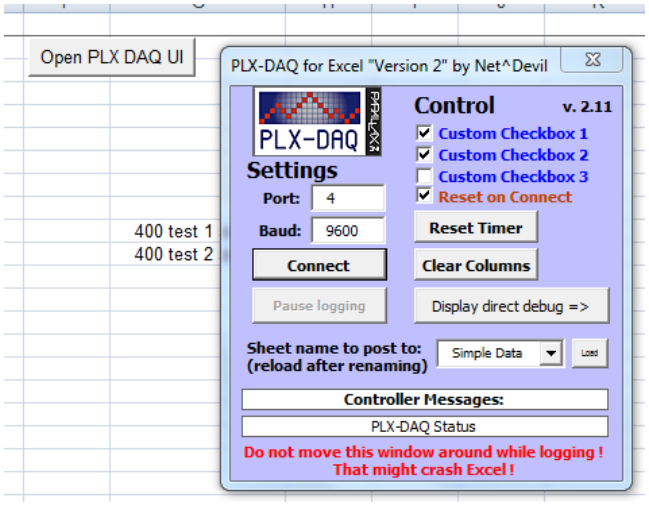
Στην συνέχεια ενεργοποιούμε το πρόγραμμα (μακροεντολές στο Excel) του PLX DAQ v2 μέσα στο αρχείο Excel που κατεβάσαμε



	A	B	C	D	E	F	G
1	Date	Time	Timer	Counter	millis		
2	7/5/17	4:06:32 PM	1.933594	0	151		
3	7/5/17	4:06:32 PM	1.996094	1	194		
4	7/5/17	4:06:32 PM	2.027344	2	238		
5	7/5/17	4:06:32 PM	2.058594	3	282		
6	7/5/17	4:06:32 PM	2.105469	4	325		
7	7/5/17	4:06:32 PM	2.167969	5	369		



Έπειτα βλέπουμε την εικόνα του προγράμματος που ανοίγει

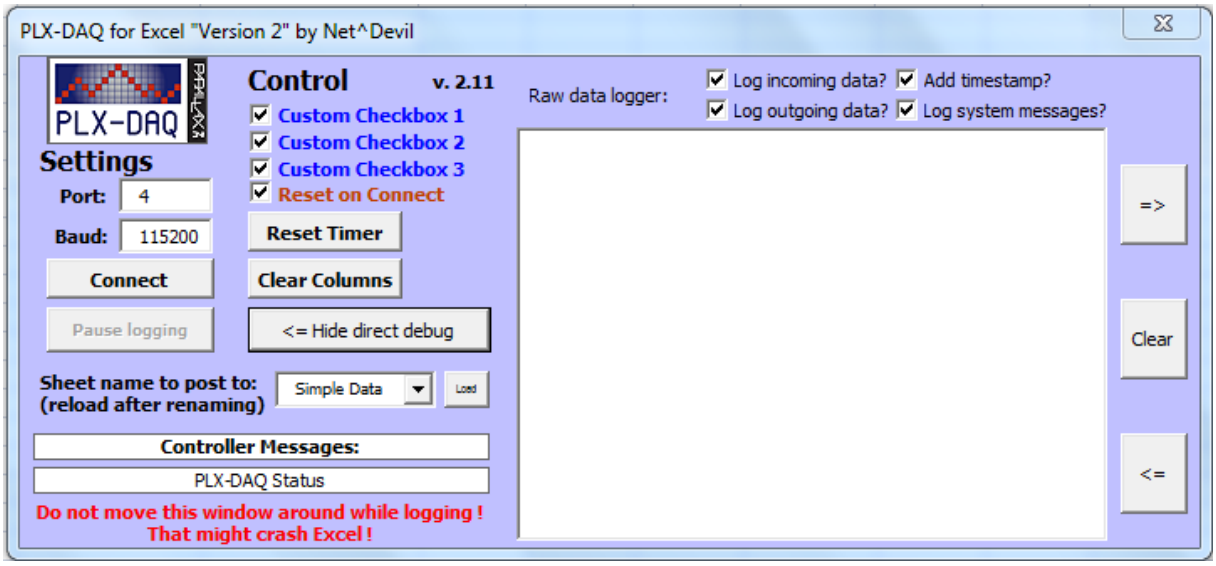


Οι βασικές ρυθμίσεις που γίνανε για να συνδεθεί σωστά με τον MSP-EXP432P401R είναι οι εξής:

- Πρώτον ορίζουμε των αριθμό θύρας (Port), που είναι συνδεμένος ο μικροελεγκτής μας

- Έπειτα ρυθμίζομαι τον ρυθμό μετάδοσης της θύρας (Baud), πρέπει να είναι ίδιος με του μικροελεγκτή
- Επίσης επιλέγουμε το Reset on Connect, τα κουτάκια των Custom Checkbox δεν θα χρησιμοποιηθούν επομένως τα αφήνουμε όπως έχει (δεν έχει σημασία αν τα έχουμε επιλέξει ή όχι)
- Στο display direct debug επιλεγούμε όλα τα κουτάκια του Raw data logger
- Στο Sheet name to post to: επιλέγουμε το Simple Data

Ακολουθεί φωτογραφία με τις ρυθμίσεις του PLX-DAQ που έγιναν για την ραφιέρα



Στην συνέχεια θα δούμε τις βασικές εντολές του PLX DAQ v2 που χρησιμοποιήθηκαν στην ραφιέρα. Αυτές οι εντολές γράφονται εμφωλευμένα μέσα στις εντολές Serial.print ή Serial.println και αποστέλλονται μέσω της σειριακής θύρας στον υπολογιστή και επικοινωνούμε με το Excel.

Οι βασικές εντολές είναι:

Η εντολή **CLEARSHEET**, αυτή η εντολή καθαρίζει όλα τα δεδομένα του ενεργού φύλλου (active sheet) του Excel (συμπεριλαμβανομένων και των ετικετών). Θα πρέπει να είναι η πρώτη εντολή σε κάθε sketch.

Η σύνταξη της είναι: Serial.println("CLEARSHEET");

Η εντολή **CLEARDATA**, αυτή η εντολή καθαρίζει μόνο τα καταγεγραμμένα δεδομένα στο ενεργό φύλλο (ξεκινώντας από τη δεύτερη γραμμή)

Η σύνταξη της είναι: Serial.println("CLEARDATA");

Η εντολή **LABEL**, με αυτήν την εντολή μπορούμε να ορίσουμε τις ετικέτες στην πρώτη γραμμή του ενεργού φύλλου στο Excel.

Η σύνταξη της είναι: Serial.println("LABEL,Πρώτη στήλη, Δεύτερη στήλη, Τρίτη στήλη, ...κλπ");

Η εντολή **DATA**, αυτή είναι η πιο βασική και κρίσιμη εντολή του PLX DAQ v2. Χρησιμοποιείται για την αποστολή δεδομένων από τον MSP-EXP432P401R στο Excel και την εκτύπωση τους στο ενεργό φύλλο του Excel.

Μπορούμε να στείλουμε οτιδήποτε θέλουμε, αλλά θα πρέπει να βεβαιωθούμε ότι έχουμε διαχωρίσει τα δεδομένα με κόμματα και ότι ο αριθμός των κομμάτων με τον αριθμό των στηλών που ορίσαμε με την εντολή LABEL είναι ο ίδιος.

Οι αποκλειστικές λέξεις κώδικα DATE, TIME και TIMER αναγνωρίζονται από το PLX DAQ και αντικαθιστούνται με τιμές.

- DATE θα αλλάξει στην τρέχουσα ημερομηνία του υπολογιστή των Windows (π.χ. 15/10/2018)
- TIME θα αλλάξει στον τρέχοντα χρόνο του υπολογιστή των Windows (π.χ. 22:38:15 ή 10:38:15 PM)
- TIMER θα αλλάξει στον χρόνο που η καταγραφή είναι ήδη ενεργή (π.χ., 1.365 δευτερόλεπτα)

Μπορούμε να προσθέσουμε τη λέξη-κλειδί AUTOSCROLL_20 στην συμβολοσειρά (String). Αυτή η πληροφορία θα κάνει το PLX DAQ να μετακινήσει αυτόματα το παράθυρο του Excel προς τα κάτω με κάθε νέα γραμμή δεδομένων που λαμβάνεται. Ο αριθμός (π.χ. 20) υποδεικνύει πόσες γραμμές πρέπει να εμφανίζονται πάνω από την τελευταία γραμμή που έχει μετακινηθεί αυτόματα.

Τρόποι σύνταξης της DATA:

- Για σταθερές πληροφορίες (Static information): Serial.println("DATA,DATE,TIME");
- Για να συμπεριλάβουμε κλήσεις συναρτήσεων ή μεταβλητών στην ίδια γραμμή: Serial.println((String) "DATA,DATE,TIME," + millis());
- Άλλος ένας τρόπος για να παρουσιάσουμε τον κώδικα με περισσότερες γραμμές αλλά με καλύτερη αναγνωσιμότητα αντί να χρησιμοποιούμαι πολλές μεταβλητές σε μία γραμμή. Χρησιμοποιούμαι μόνο την εντολή print, καθώς το println χρησιμοποιείται μόνο στο τέλος για να στείλει την συμβολοσειρά πλήρη, παράδειγμα:

Serial.print("DATA,DATE,TIME,");

Serial.print(H πρώτη μου μεταβλητή);

Serial.print(",”); //αλλάζουμε στήλη με κάθε κόμμα

Serial.print(millis());//κλήση μιας συνάρτησης

Serial.print(",”); //ξανά αλλαγή στήλης

Serial.println(H δεύτερη μου μεταβλητή);

Ακολουθεί φωτογραφία του κώδικα που μεταφέρει τις μετρήσεις της ραφιέρας στο Excel.

```
246   //Katagrafi sto Excel (prepei na brisketai sto idio path me to programma tou Slave
247   if ((digitalRead(pinakiExcel) == HIGH) && (b == 0))
248   {
249     Serial.println("CLEARSHEET"); // ka8arizi to fullo ergasias tou excel kai 3ekinaei stin grammi 1
250     Serial.println("LABEL,Date,Time,Material Name,Quantity,Position at shelf"); //ka8orizo tous titlous sto excel
251     for (i = 0; i < 50; i++)
252     {
253       Serial.print((String) "DATA,DATE,TIME,");
254       Serial.print(pinNames[i]);
255       Serial.print(",");
256       Serial.print(pinQuantity[i]);
257       Serial.print(",");
258       Serial.println((i + 1));
259     }
260     b = 1; //Flag gia na tre3i mia fora mono h katagrafi se ka8e pathma tou koumpiou
261   }
262   if (digitalRead(pinakiExcel) == LOW)
263   {
264     b = 0;
265   }
```

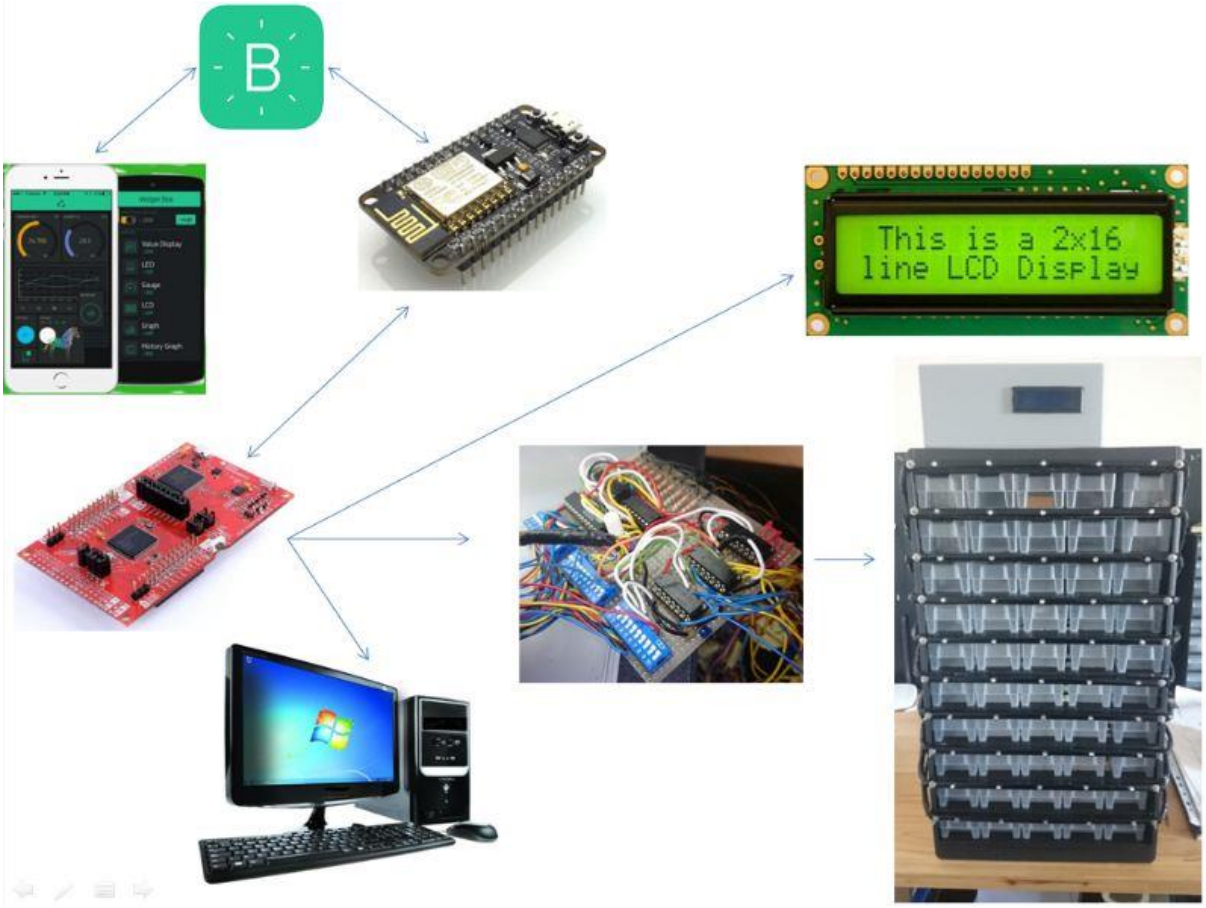
και πως μεταφέρονται τελικά στο Excel (είναι η Εικόνα 60).

	A	B	C	D	E
1	Date	Time	Material Name	Quantity	Position at shelf
2	15/10/18	10:38:15 PM	1. 7400	1	1
3	15/10/18	10:38:15 PM	2. 7400	2	2
4	15/10/18	10:38:15 PM	3. 7400	3	3
5	15/10/18	10:38:15 PM	4. 7400	4	4
6	15/10/18	10:38:15 PM	5. 7400	5	5
7	15/10/18	10:38:15 PM	6. 7400	6	6
8	15/10/18	10:38:15 PM	7. 7400	7	7
9	15/10/18	10:38:15 PM	8. 7400	8	8
10	15/10/18	10:38:15 PM	9. 7400	9	9
11	15/10/18	10:38:15 PM	10. 7400	10	10
12	15/10/18	10:38:15 PM	11. 7400	11	11
13	15/10/18	10:38:15 PM	12. 7400	12	12
14	15/10/18	10:38:15 PM	13. 7400	13	13
15	15/10/18	10:38:15 PM	14. 7400	14	14
16	15/10/18	10:38:15 PM	15. 7400	15	15
17	15/10/18	10:38:15 PM	16. 7400	16	16
18	15/10/18	10:38:15 PM	17. 7400	17	17
19	15/10/18	10:38:15 PM	18. 7400	18	18
20	15/10/18	10:38:15 PM	19. 7400	19	19
21	15/10/18	10:38:15 PM	20. 7400	20	20
22	15/10/18	10:38:15 PM	21. 7400	21	21

Λειτουργία ραφιέρας, εφαρμογής ταξινόμησης υλικών σε ράφια

Οι εφαρμογή ταξινόμησης υλικών σε ράφια μας δίνει την δυνατότητα μέσω του κινητού μας να κάνουμε αναζήτηση οποιουδήποτε υλικού έχουμε καταχώρηση στην ραφιέρα μας, να του αλλάξουμε την ποσότητα, να καταγράφουμε τις μετρήσεις (ημερομηνία και ώρα καταγραφής, όνομα, ποσότητα και θέση υλικού) τις ραφιέρας στο Microsoft Excel και τέλος να κάνουμε γενικό έλεγχο της ραφιέρας (τρέχει σειριακά όλα τα RGB LED).

Ο τρόπος λειτουργίας της ραφιέρας περιγράφεται στο μπλοκ διάγραμμα της Εικόνας 61.



Εικόνα 61. Μπλοκ διάγραμμα λειτουργίας της ραφιέρας

Αναλυτικά βλέπουμε στην Εικόνα 61. , ξεκινώντας από πάνω αριστερά:

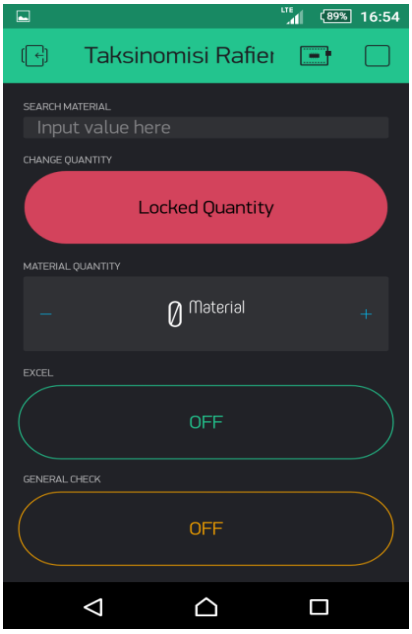
- Μέσω των εργαλείων (Widgets), της εφαρμογής Blynk που έχουμε εγκαταστήσει στο κινητό μας τηλέφωνο, στέλνουμε πληροφορίες (εντολές) στους Blynk Servers μέσω WiFi

- Έπειτα από τους Blynk Servers στέλνονται οι πληροφορίες στον ESP8266 που είναι συνδεμένος στο Internet μέσω WiFi
- Στην συνέχεια εφόσον ο ESP8266 έχει λάβει τις εντολές για το πώς θα λειτουργήσει η ραφιέρα, επεξεργάζεται κατάλληλα τα δεδομένα που πείρε (σύμφωνα με το πρόγραμμα που φτιάχτηκε για τον ESP8266, βλέπε παράρτημα A Master_ESP8266_Rafiera) και συνδέεται μέσω του SPI με τον MSP-EXP432P401R για να τα μεταδώσει, στέλνοντας ένα Byte πληροφορίας κάθε φορά, ο ρόλος του ESP8266 είναι ο Master στην SPI επικοινωνία. Επίσης ο ESP8266 εκτός του SPI έχει 3 pins εξόδους συνδεμένα στον MSP-EXP432P401R που του δίνουν επιπλέον οδηγίες για το πως θα λειτουργήσει η ραφιέρα, έχει 1 pin επίτρεψης για την αλλαγή ποσότητας των υλικών, 1 pin επίτρεψης για την καταγραφή υλικών στο Microsoft Excel και τέλος 1 pin επίτρεψης για τον γενικό έλεγχο της ραφιέρας.
- Ο MSP-EXP432P401R με την σειρά του σαν Slave λαμβάνει όλα τα δεδομένα που αποστέλλονται από το SPI και τα 3 pins του ESP8266, τα επεξεργάζεται κατάλληλα (σύμφωνα με το πρόγραμμα που φτιάχτηκε για τον MSP-EXP432P401R, βλέπε παράρτημα A Slave_msp432_Rafiera) και στέλνει πληροφορίες στα εξής:
    - Στην πλακέτα των αποκωδικοποιητών/ αποπλεκτών (74155, 74154) για τον έλεγχο των RGB LED της ραφιέρας (Θέση και Χρώμα)
    - Στην LCD οθόνη για να βλέπουμε τις ενδείξεις του κάθε ραφιού που επιλέγεται (Όνομα, Ποσότητα, Θέση)
    - Στον υπολογιστή μέσω της σειριακής θύρας για την καταγραφή των μετρήσεων της ραφιέρας στο Microsoft Excel

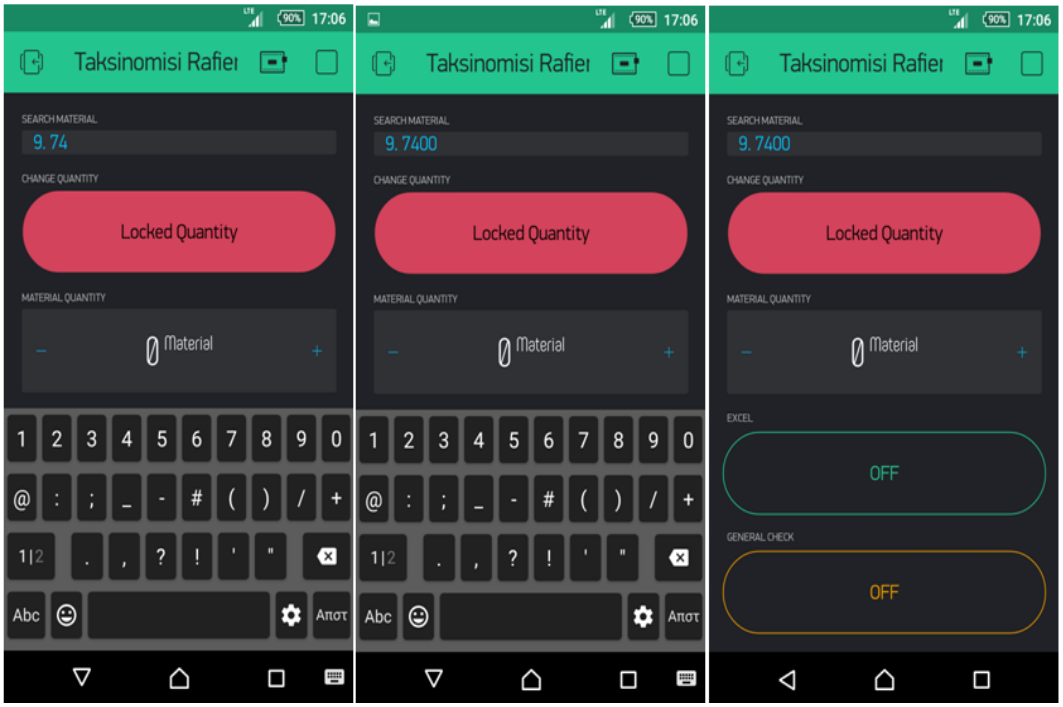
Επίσης όταν τροφοδοτήσουμε την ραφιέρα μας μέσω των 2 USB βυσμάτων των μικροελεγκτών για να αρχίσει να λειτουργεί, απαιτείται Reset των 2 μικροελεγκτών ταυτόχρονα, για να συγχρονιστούν οι μικροελεγκτές, διότι ενώνονται μεταξύ τους μέσω του SPI.

Ακολουθεί αναλυτικά βήμα προς βήμα η λειτουργία της εφαρμογής ταξινόμησης υλικών σε ράφια με φωτογραφίες εν ώρα λειτουργίας της ραφιέρας, τα εργαλεία που χρησιμοποιήθηκαν στην εφαρμογή αναλύθηκαν στο υποκεφάλαιο της Εφαρμογής Blynk.

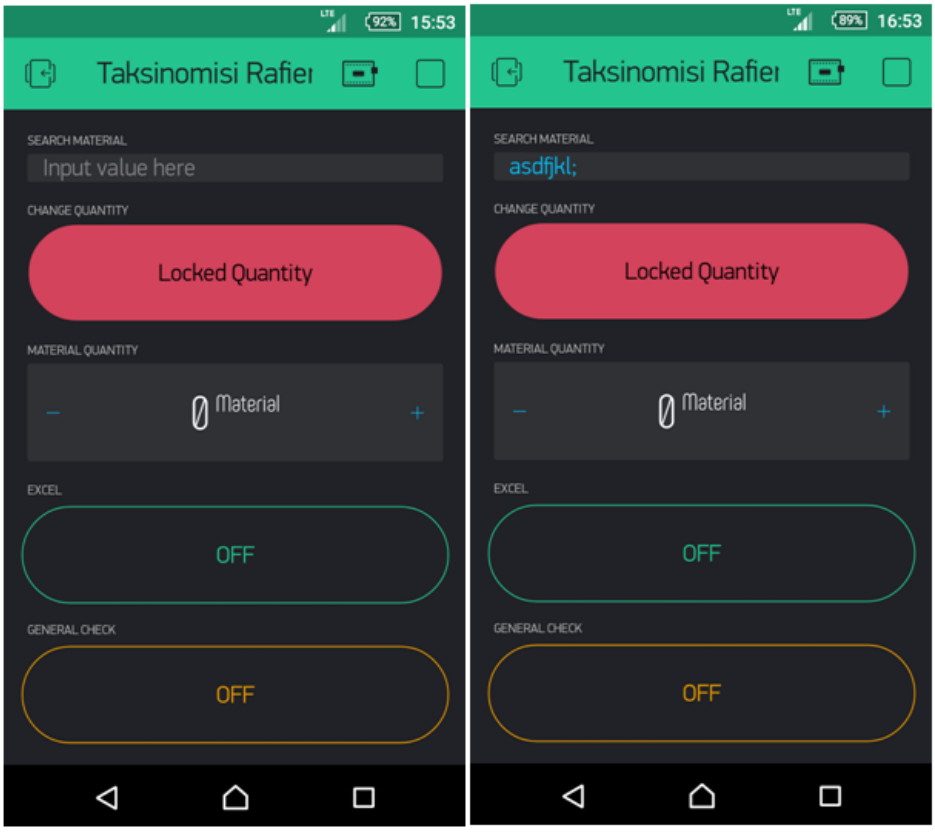
Είδαμε στην Εικόνα 59. το γραφικό περιβάλλον της εφαρμογής μας, φαίνεται και παρακάτω εν ώρα λειτουργίας:



1. Στην αρχή βλέπουμε το πρώτο εργαλείο (widget) της εφαρμογής πού είναι το text input με την ονομασία Search Material, γράφοντας σε αυτό την ονομασία του υλικού προς αναζήτηση, αν το όνομα υπάρχει περασμένο στους πίνακες που είναι καταχωρημένα τα ονόματα των υλικών της ραφιέρας ανάβει το RGB LED στην θέση του προς αναζήτηση υλικού και στον κατάλληλο χρωματισμό (εξαρτάται από την ποσότητα του υλικού), καθώς βλέπουμε και αναλυτικά όλες οι πληροφορίες του υλικού προς αναζήτηση στην LCD οθόνη. Ακολουθούν φωτογραφίες χρήσης του Search Material με ονομασία που υπάρχει στο πίνακα καταχωρήσεων ονομάτων της ραφιέρας.

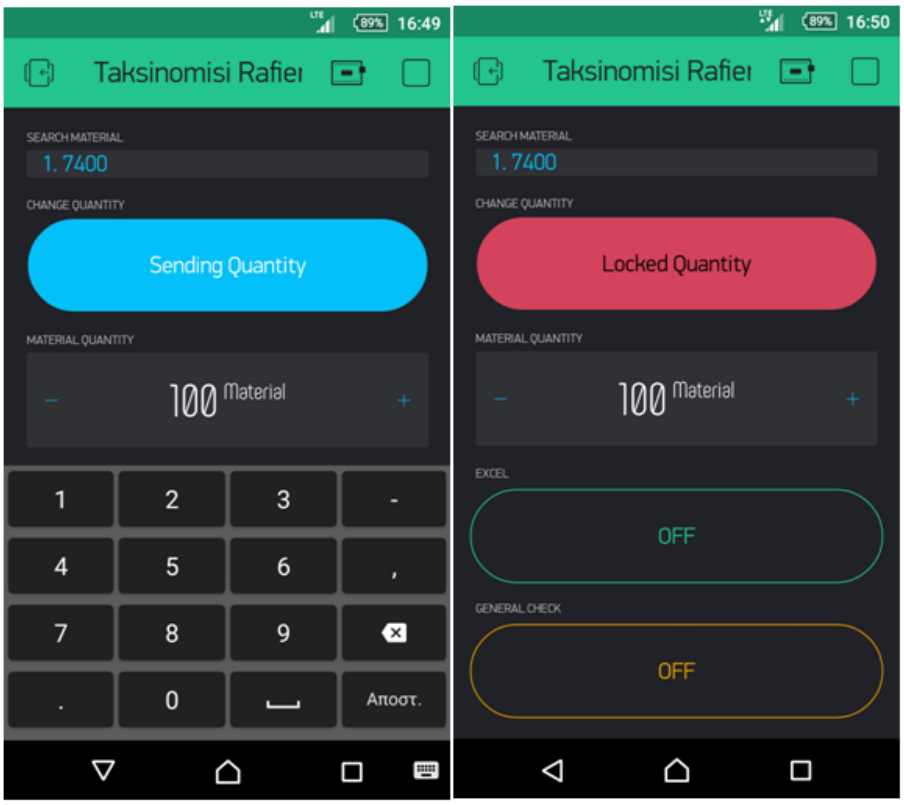


2. Σε περίπτωση που δεν γράψεις κάτι στο Search Material ή γράψεις κάτι που δεν υπάρχει στους πίνακες καταχωρήσεων ονομάτων της ραφιέρας μας βγάζει το μήνυμα Name: Not found search again... , δηλαδή ότι δεν βρήκε το υλικό και πρέπει να κάνεις εκ νέου αναζήτηση.



3. Το δεύτερο (Styled Button) και τρίτο εργαλείο (Numeric Input) έχουν τις ονομασίες Change Quantity και Material Quantity αντίστοιχα που βλέπουμε στην εφαρμογή μας, χρησιμοποιούνται για να αλλάζουμε την ποσότητα του επιλεγμένου υλικού, το Change Quantity πρέπει να πατηθεί για να γίνει η επίτρεψη αλλαγής ποσότητας του υλικού, όσο μένει πατημένο στέλνει συνέχεια την ποσότητα που ρυθμίζεται από το

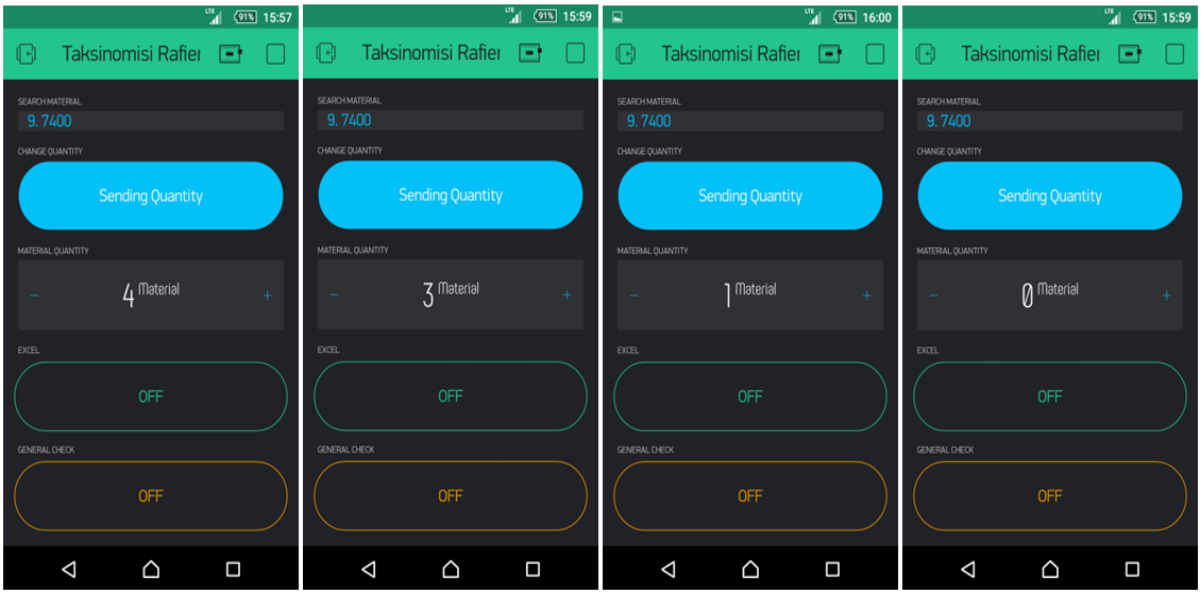
Material Quantity, το Material Quantity μας δίνει την δυνατότητα ρύθμισης της ποσότητας του υλικού από 0 έως 255



4. Επίσης ο χρωματισμός των RGB LED εξαρτάται από την ποσότητα του υλικού σε κάθε ράφι, ως εξής:
    a. αν το RGB LED γίνει κόκκινο σημαίνει ότι έχει 0 υλικά (Materials) το ράφι
    b. αν γίνει μπλέ σημαίνει ότι έχει από 1 έως 3 υλικά
    c. τέλος αν γίνει πράσινο έχει από 4 υλικά και πάνω

Παράδειγμα αλλαγής ποσότητας στο ράφι νούμερο 9, στην αρχή είχε 9 υλικά και άναβε πράσινο.

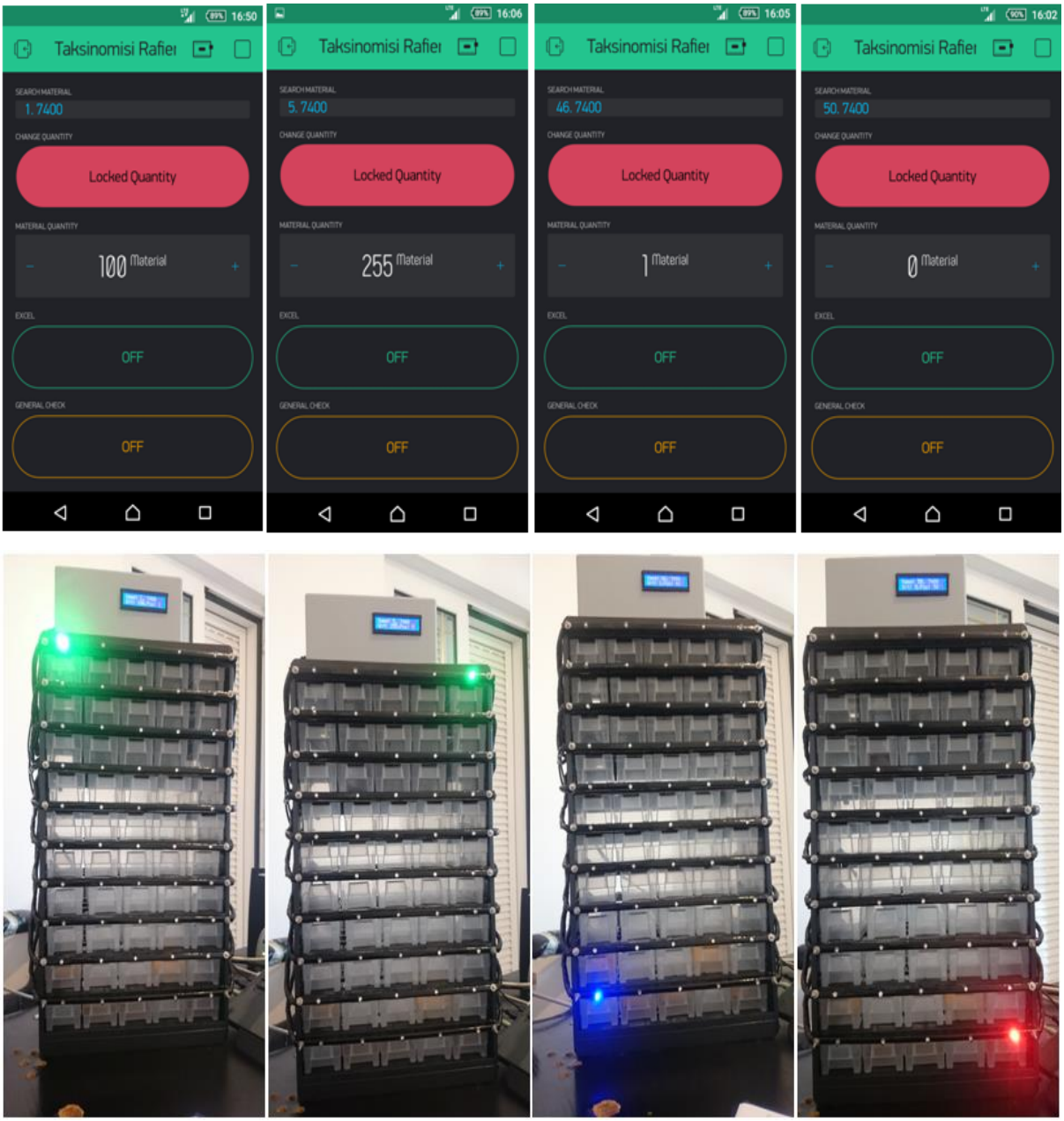
Στην συνέχεια του αλλάζουμε την ποσότητα από 9 σε 4, 3, 1, 0 και παρατηρούμε τις αλλαγές του χρωματισμού του RGB LED:




Name: 9. 7400
Qnt: 4,Pos: 9

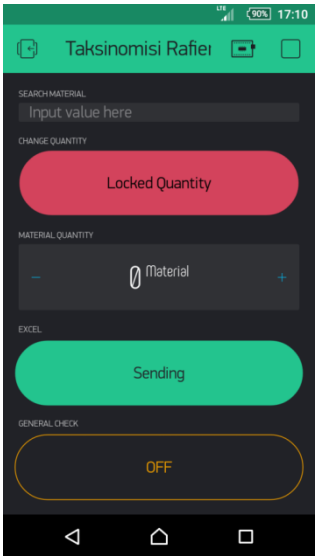

Name: 9. 7400
Qnt: 3,Pos: 9

5. Επίσης οι θέσεις των ραφιών/συρταριών είναι 50 και αντιστοιχεί 1 RGB LED στο κάθε ένα ράφι. Τα RGB LED βρίσκονται μέσα σε κανάλια, κάθε κανάλι αποτελείται από 5 RGB LED. Η αρίθμηση των θέσεων της ραφιέρας ξεκινάει από πάνω αριστερά που είναι η θέση 1, προς τέρμα δεξιά που είναι η θέση 5, έπειτα ακριβώς κάτω από την θέση 1, είναι η θέση 6 και τέρμα δεξιά στο ίδιο κανάλι είναι η θέση 10 και συνεχίζει έτσι μέχρι τέλους στην θέση 50 που βρίσκεται τέρμα κάτω δεξιά της ραφιέρας, αποτελείται από 10 κανάλια συνολικά. Ακολουθούν φωτογραφίες από τις θέσεις των άκρων της ραφιέρας, θέσεις: 1, 5, 46 και 50.

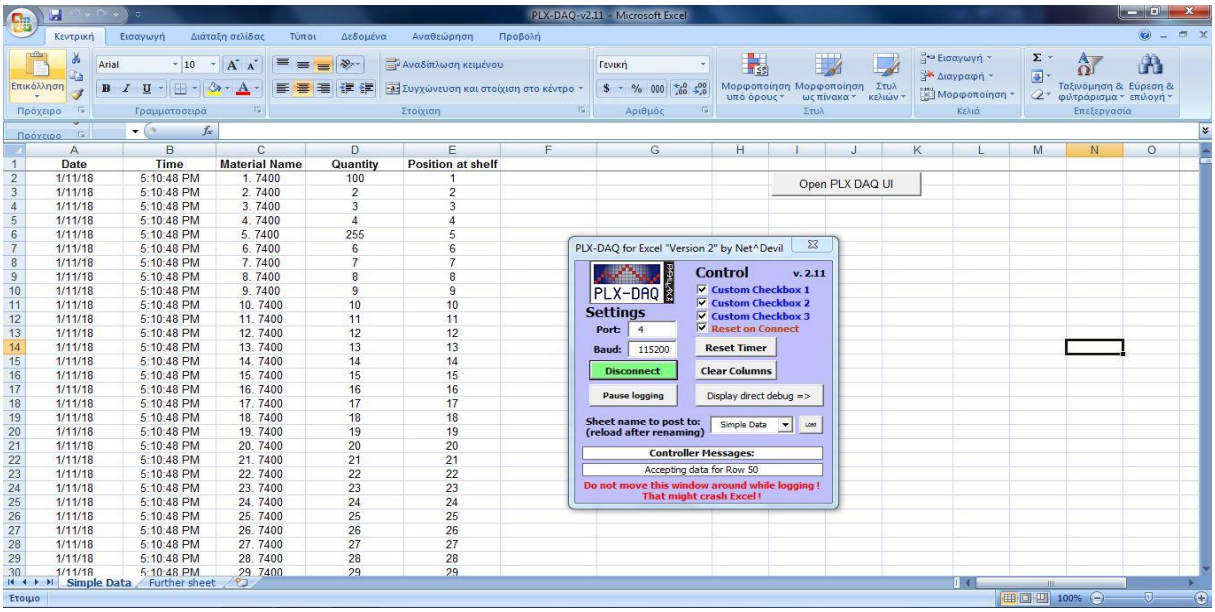


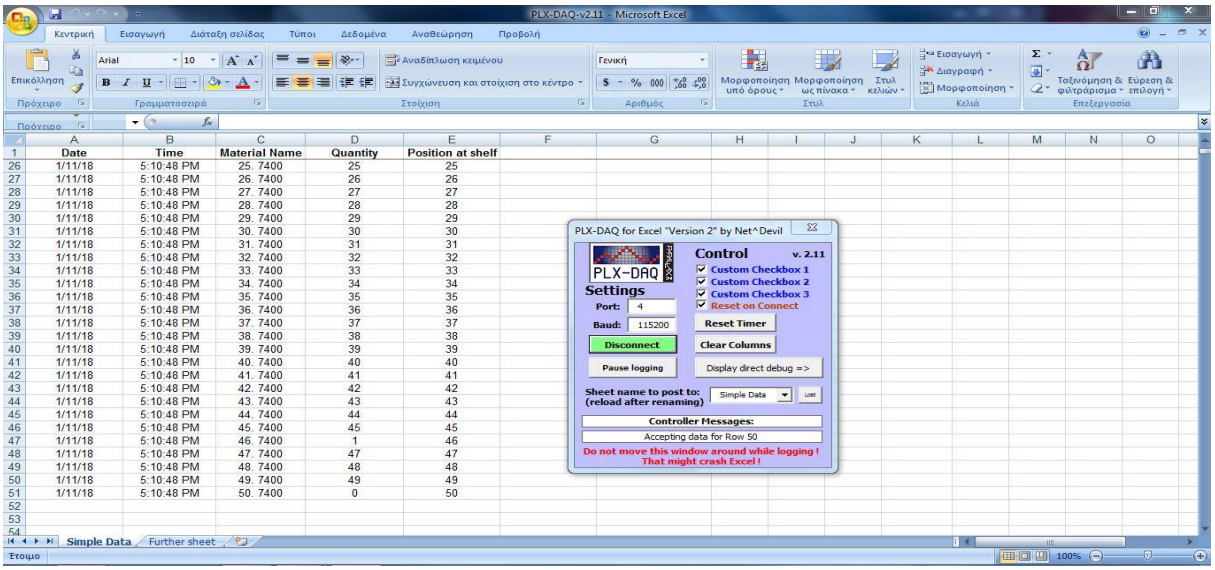
6. Ακολουθεί έπειτα το τέταρτο εργαλείο (απλό Button) της εφαρμογή μας, με την ονομασία Excel. Με το που πατηθεί γίνεται η καταγραφή το τρέχων δεδομένων της ραφιέρας στο Microsoft Excel και βλέπουμε το μήνυμα Sending. Τα δεδομένα που αποθηκεύονται είναι:
    a. η ημερομηνία (Date) και η ώρα (Time) της καταγραφής των δεδομένων
    b. τα ονόματα των υλικών (Material Name) που υπάρχουν στα ράφια
    c. η ποσότητα του κάθε υλικού (Quantity)
    d. και τέλος οι θέσεις των υλικών στα ράφια (Position at shelf)

Ξαναπατάμε το κουμπί για να ελευθερωθεί η εφαρμογή και να γράφει OFF το κουμπί.

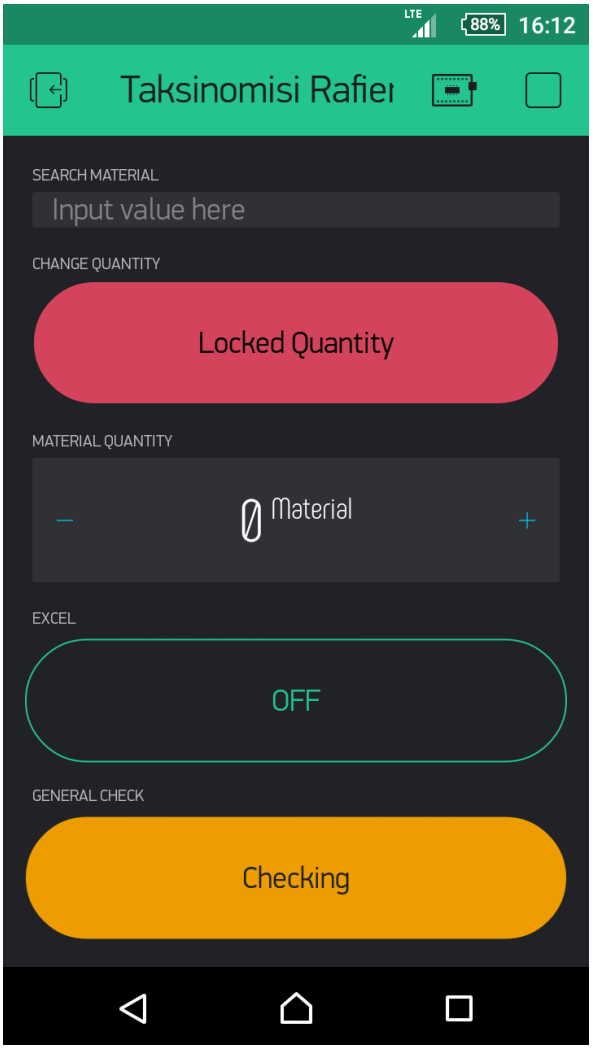


	A	B	C	D	E
1	Date	Time	Material Name	Quantity	Position at shelf





7. Το τελευταίο εργαλείο της εφαρμογής μας (απλό Button) έχει την ονομασία General Check και με το που πατηθεί ελέγχει σειριακά ένα προς ένα τα ράφια ξεκινώντας από το ράφι νούμερο 1 φτάνοντας στο ράφι νούμερο 50, ανάβοντας τα RGB LED στον κατάλληλο χρωματισμό αναλόγως την ποσότητα υλικών που διαθέτει το κάθε ράφι. Το RGB LED του κάθε ραφιού ανάβει για 400 mSeconds, συνολικά το κάθε κανάλι ελέγχεται μέσα σε 2 Seconds γιατί διαθέτει 5 RGB LED, ο γενικός έλεγχος τελειώνει μετά από 20 Seconds, διότι η ραφιέρα διαθέτει 10 κανάλια. Επίσης όσο το κουμπί του γενικού ελέγχου είναι πατημένο στην εφαρμογή μας δείχνει το μήνυμα Checking και η LCD οθόνη μας δείχνει το μήνυμα: "General check in progress..." και κάνει συνεχόμενο έλεγχο μέχρι να ξανά πατηθεί άλλη μια φόρα για γράψει OFF, όταν γράψει OFF συνεχίζει των έλεγχο της ραφιέρας μέχρι τέλους και σταματάει τον γενικό έλεγχο και αναμένει για την επομένη εντολή.

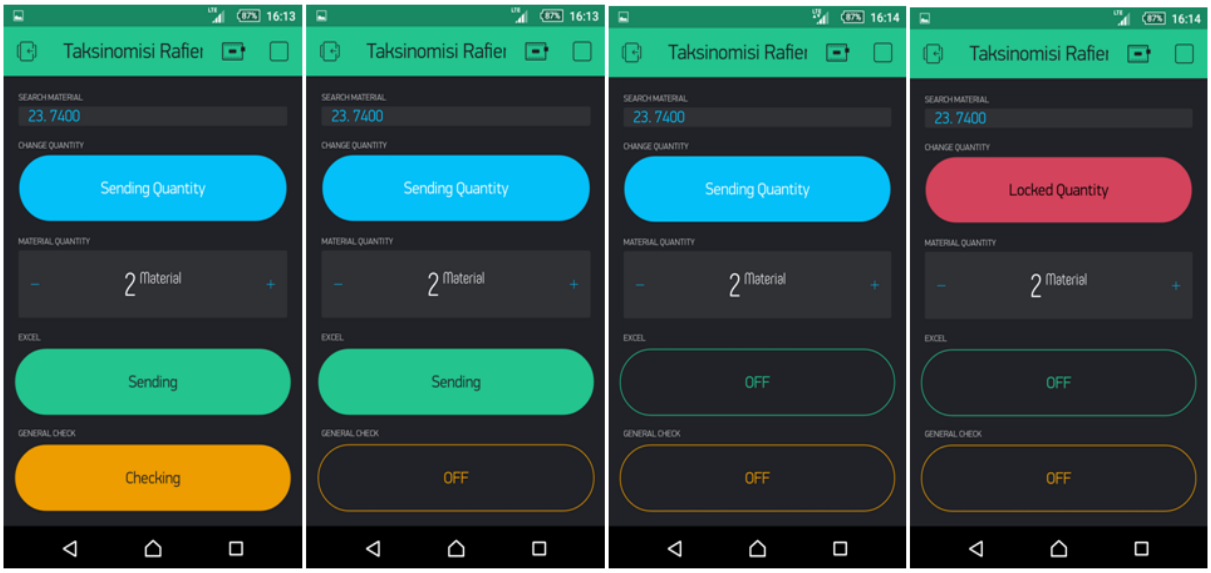



General check
in Progress...

8. Η προτεραιότητα των εργαλείων (widgets) της εφαρμογής ταξινόμησης υλικών σε ράφια είναι η εξής:

   General Check > Excel > Change Material Quantity > Search Material

   Δηλαδή όσο λειτουργεί το General Check δεν μπορείς να χρησιμοποιήσεις κανένα άλλο εργαλείο, έπειτα εφόσον κλείσεις το General Check τρέχει αμέσως το Excel και γίνεται η καταγραφή, όσο είναι πατημένο το Excel δεν γίνεται καμία άλλη λειτουργία εκτός αν πατηθεί το General Check που έχει μεγαλύτερη προτεραιότητα, έπειτα γίνεται η αλλαγή της ποσότητας του επιλεγμένου υλικού, ισχύει πάλι η προτεραιότητα, τέλος για να γίνει εκ νέου αναζήτηση υλικού πρέπει να είναι όλα τα άλλα εργαλεία κλειστά. Ακολουθούν φωτογραφίες επαλήθευσης της προτεραιότητας.



Πρώτα βλέπουμε ότι κάνει γενικό έλεγχο (General Check)

Στην συνέχεια σβήνοντας (OFF) το κουμπί του General Check, κάνει καταγραφή στο Microsoft Excel της τρέχων τιμές της ραφιέρας, ενδιαφέρον είναι να δούμε το ράφι με το νούμερο 23, η τιμή της ποσότητας υλικών είναι 23 και του ζητάμε να γίνει 2 αλλά λόγο προτεραιότητας γίνεται πρώτα η καταγραφή του Excel πριν προλάβει να αλλάξει σε 2.

24	1/11/18	4:13:19 PM	23. 7400	23	23

Σβήνοντας (OFF) το κουμπί του Excel βλέπουμε ότι γίνεται τελικά και η αλλαγή της ποσότητας του ραφιού στην ζητούμενη ποσότητα 2 από το 23 που ήταν πριν.

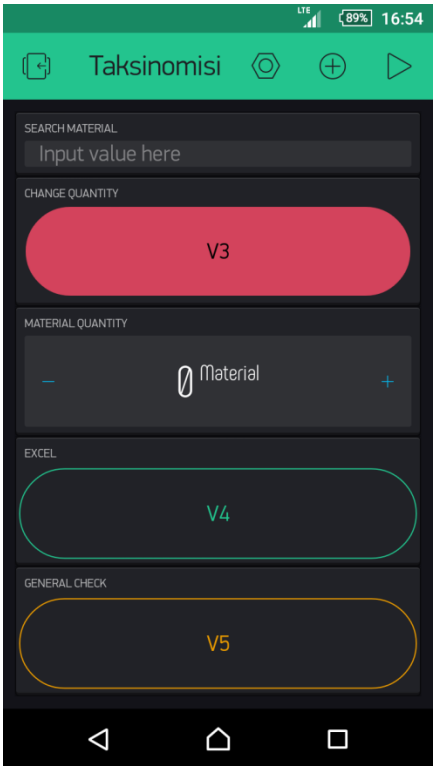
Τώρα για δοκιμή ξαναπατήθηκε το κουμπί του Excel για να δούμε την νέα καταγραφή της ποσότητας στο Microsoft Excel,

24	1/11/18	4:17:20 PM	23. 7400	2	23

όντος έγινε η καταγραφή με την νέα ποσότητα του ραφιού 23 που είναι 2 υλικά.

Τέλος με σβηστά όλα τα εργαλεία μπορούμε να κάνουμε νέα αναζήτηση υλικού.

Στην συνέχεια πατώντας το κουμπί Stop στην εφαρμογή του Blynk η εφαρμογή μας σταματάει να λειτουργεί και φαίνεται κάπως έτσι:



Αν πατηθεί το κουμπί του play αρχίζει να λειτουργεί και πάλι, η ραφιέρα εκτελεί πάντα την τελευταία εντολή που της δόθηκε και ας έχει κλείσει η εφαρμογή στο κινητό.

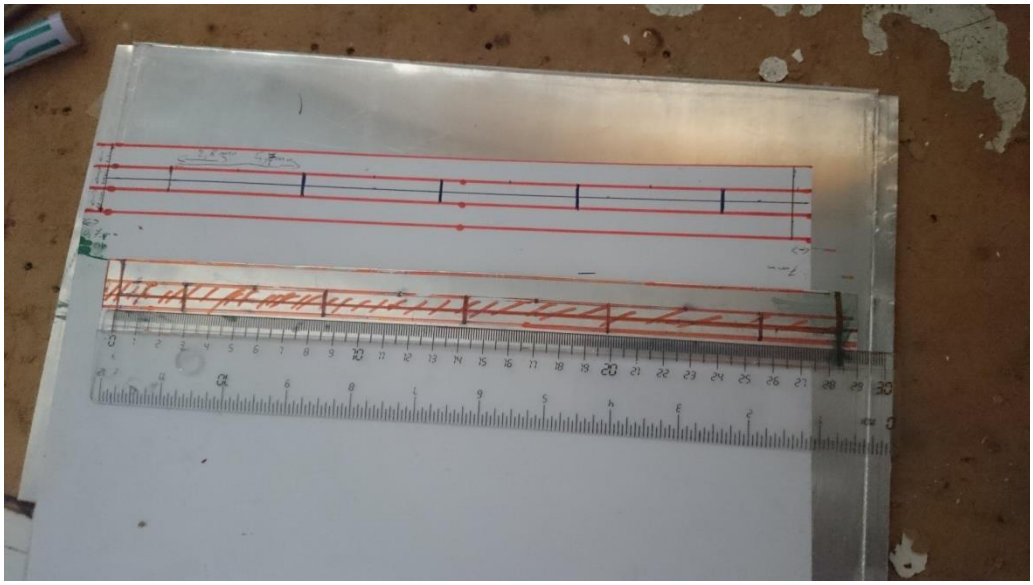
Κατασκευή ραφιέρας, μετατροπή απλής ραφιέρας σε έξυπνη ραφιέρα

Ακολουθεί η περιγραφή της κατασκευής της ραφιέρας, η μετατροπή της από απλή ραφιέρα σε έξυπνη, θα περιγραφτεί παρακάτω βήμα προς βήμα.

1. Πρώτα πήραμε μια απλή ραφιέρα μεγέθους 50 θέσεων, 10 γραμμές x 5 στήλες και ξεκινήσαμε την μετατροπή.

2. Στην αρχή πάρθηκαν φύλλα αλουμινίου για την δημιουργία καναλιών που θα γίνουν η βάσεις για τα RGB LED μας, το κάθε κανάλι αποτελείται από 5 RGB LED που αντιστοιχεί 1 RGB LED σε κάθε ράφι, 1 κανάλι αντιστοιχεί σε κάθε γραμμή της ραφιέρας άρα χρειαστήκαμε συνολικά 10 κανάλια.

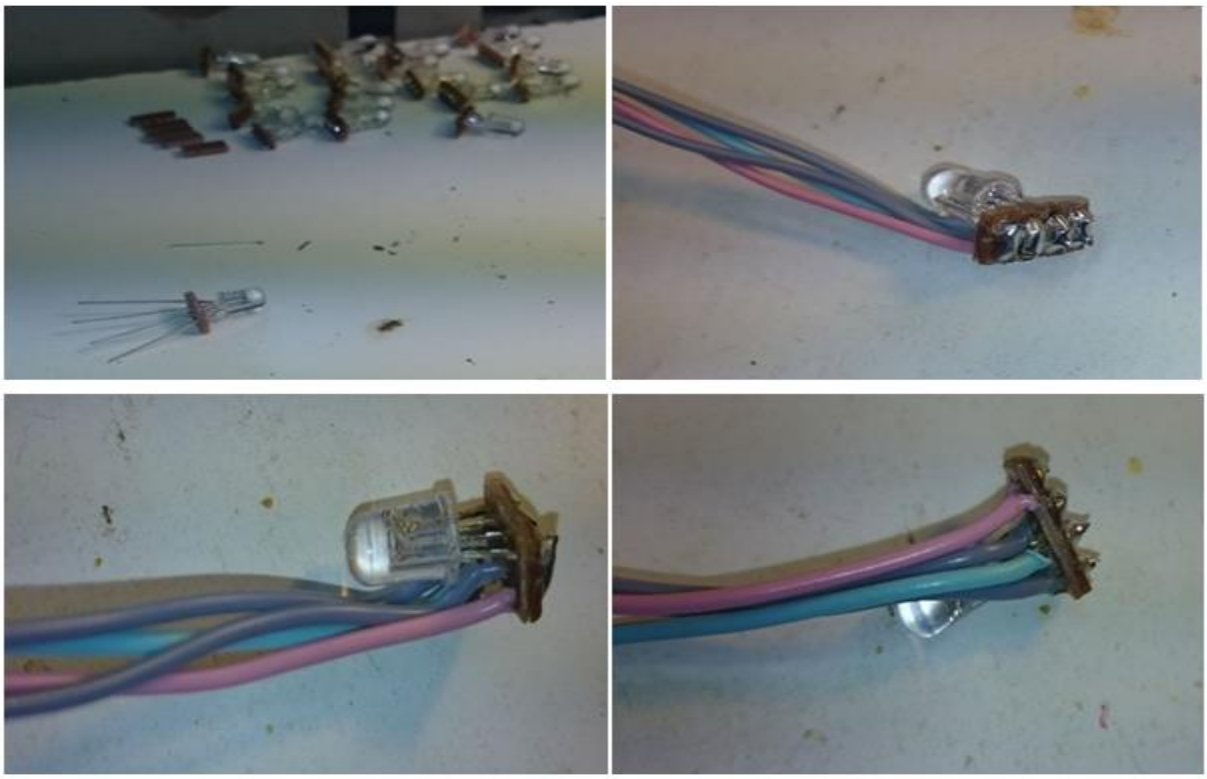
Ακολουθεί μια φωτογραφία, που γίνονταν ο σχεδιασμός των αποστάσεων πάνω στο κομμένο αλουμίνιο πριν πάει για στραντζάρισμα και τρύπημα.



3. Τα φύλλα αλουμινίου κοπήκαν, στραντζαρίστηκαν (λυγίστηκαν ώστε να πάρουν το σχήμα "Π" καναλιού) ώστε να εφαρμοστούν ακριβώς πάνω από τα ράφια που αντιπροσωπεύουν.
4. Εφόσον τα φύλλα αλουμινίου πήραν το σχήμα "Π" που θέλουμε, ανοίχτηκαν οι τρύπες υποδοχείς των RGB LED, 5 σε κάθε κανάλι, σε 5 ίσα διαστήματα μεταξύ τους, επίσης ανοίχτηκαν και 2 τρύπες για της βίδες στήριξης του καναλιού.
5. Καθώς δημιουργήθηκαν όλες οι τρύπες, σε όλα τα κανάλια, έγινε ο καθαρισμός των καναλιών από τα ρινίσματα, έπειτα περάστηκε διαλυτικού νίτρου για των καθαρισμό τυχών υπολειμμάτων λαδιού (από το τρυπάνι), λίπους από τα χέρια μας που πιάσανε τα φύλλα αλουμινίου κατά την επεξεργασία τους, χρώμα μαρκαδόρου που χρησιμοποιήθηκε για την σχεδίαση και άλλες βρώμιες που μπορεί να είχαν επάνω.
6. Στην συνέχεια μετά των καθαρισμό των καναλιών έγινε η βαφή τους με μαύρο χρώμα αυτοκινήτου.
7. Όσο στέγνωνε το χρώμα στα κανάλια, ανοίχτηκαν οι τρύπες στήριξης των καναλιών πάνω στην ραφιέρα (καθαρίστηκαν και αυτές από ρινίσματα).
8. Εφόσον ήταν έτοιμα τα κανάλια τον LED, τοποθετήσαμε τα RGB LED στα κανάλια, το κάθε LED τοποθετήθηκε πρώτα σε μια διάτρητη πλακέτα κατασκευών γενικής

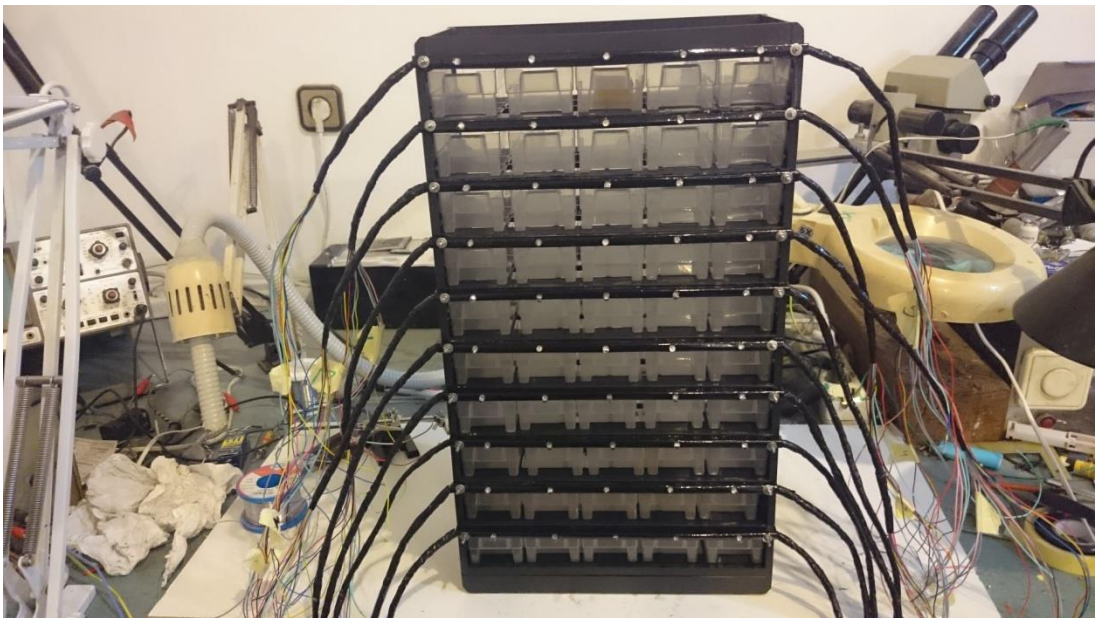
χρήσης και έπειτα συνδέθηκαν και τα 4 καλώδια τροφοδοσίας του LED πάνω σε αυτή, τα RGB LED μας είναι κοινής ανόδου (+) άρα είχαμε ένα καλώδιο για την άνοδο και τρία για τις τρείς καθόδους (-) που αντιστοιχούν σε κάθε χρώμα Κόκκινο (Red), Πράσινο (Green) και Μπλέ (Blue).

Ακολουθούν φωτογραφίες με τα RGB LED πάνω στην πλακέτα γενικής χρήσης.



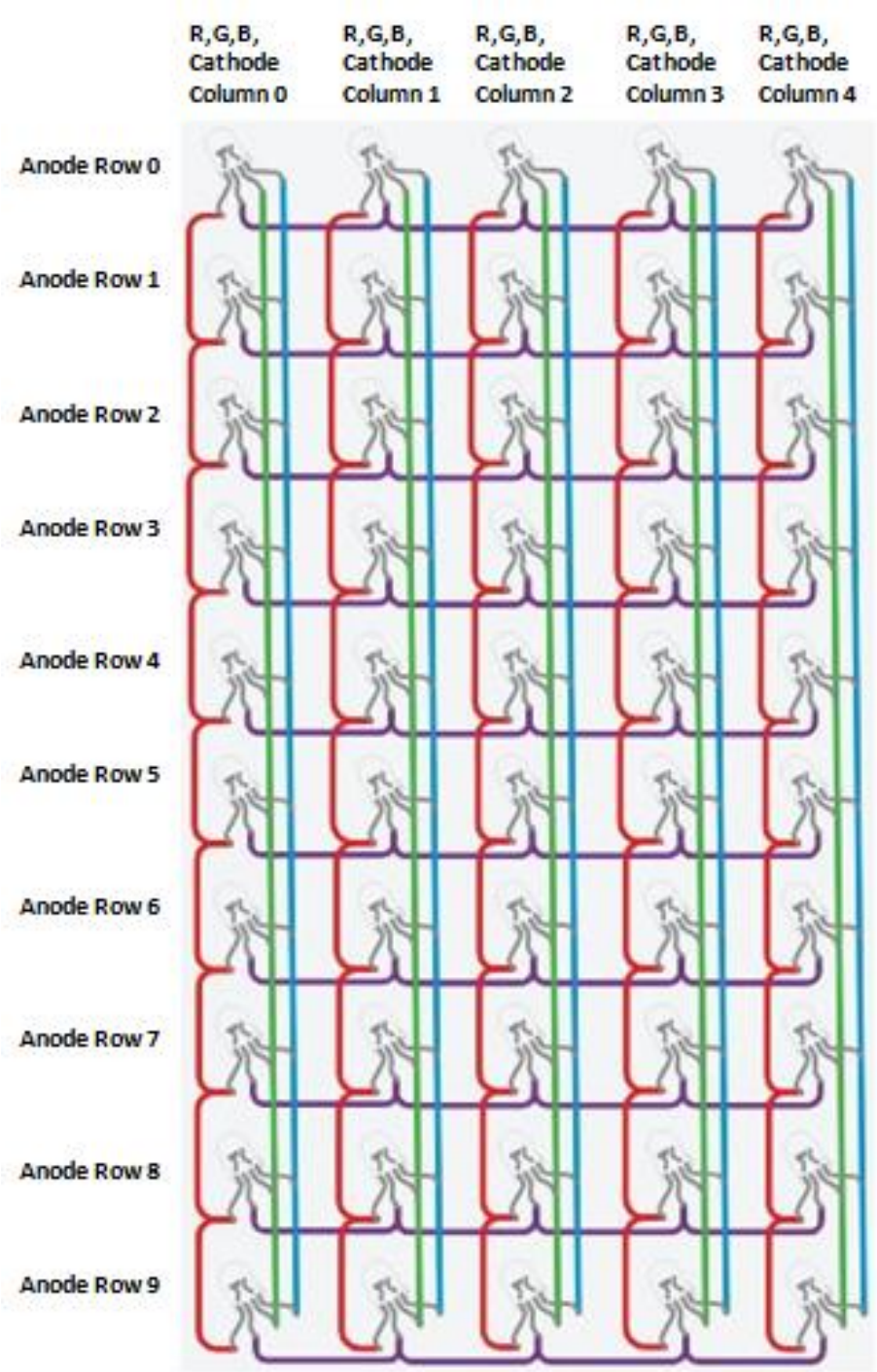
Η πλακέτα γενικής χρήσης χρησιμοποιήθηκε για να αυξηθεί η μηχανική αντοχή των ακροδεκτών των RGB LED.

9. Στην συνέχεια προσαρμόστηκαν τα κανάλια με τα RGB LED πάνω στην ραφιέρα.



10. Έπειτα έγινε η σύνδεση του πλέγματος των RGB LED, έτσι ώστε να έχουμε την δυνατότητα εύρεσης του επιθυμητού ραφιού με την χρήση συντεταγμένων, το πλέγμα το βάλαμε σε προστατευτικό κουτί πίσω από την ραφιέρα, όπως φαίνεται και στις 2 φωτογραφίες που ακολουθούν.

Στην συνέχεια ακολουθεί η Εικόνα 62. που μας δείχνει πώς συνδέθηκε το πλέγμα.



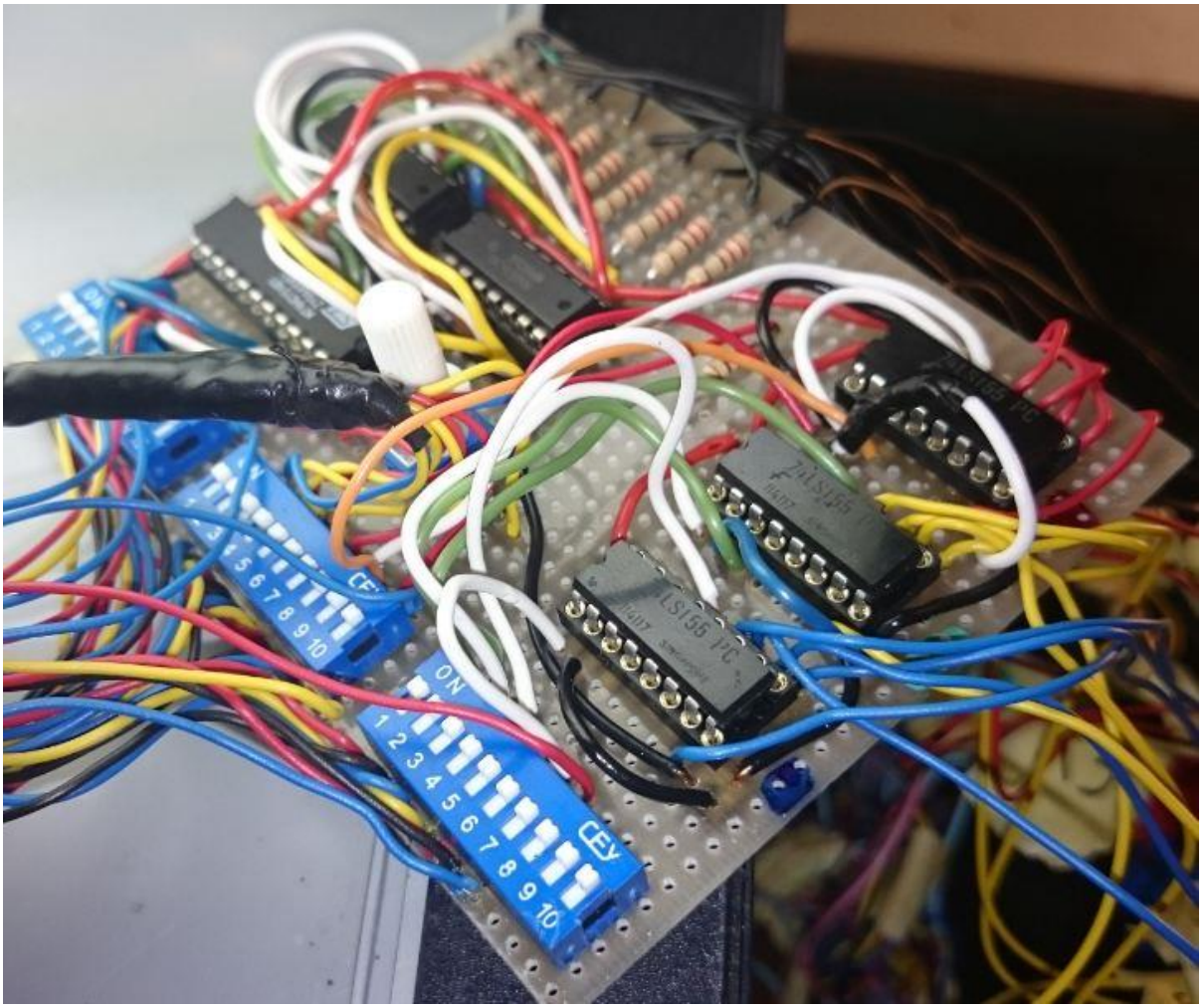
Εικόνα 62. Πλέγμα συνδέσεων των RGB LED

11. Ο κυρίως λόγος δημιουργήσαμε το πλέγμα είναι για να μειώσουμε τις συνδέσεις, διότι έχουμε 50 RGB LED που σημαίνει ότι έχουμε 200 συνδέσεις (50 RGB LED x 4 ακροδέκτες το καθένα), άρα δεν μας φτάνουν τα pin του μικροελεγκτή μας που είναι μόνο 40 προς χρήση, με την χρήση του πλέγματος από 200 συνδέσεις φτάνουμε μόνο στις 25 συνδέσεις που είναι οι εξής:

a. 10 συνδέσεις γραμμών (Rows από 0 έως 9), είναι για την κοινή άνοδο τον RGB LED, φαίνονται με το μωβ χρώμα στην Εικόνα 62.
b. 5 συνδέσεις κόκκινων στηλών (Red Columns από 0 έως 4), είναι για το κόκκινο χρώμα των RGB LED, φαίνονται με το κόκκινο χρώμα στην Εικόνα 62.
c. 5 συνδέσεις πράσινων στηλών (Green Columns από 0 έως 4), είναι για το πράσινο χρώμα των RGB LED, φαίνονται με το πράσινο χρώμα στην Εικόνα 62.
d. 5 συνδέσεις μπλέ στηλών (Blue Columns από 0 έως 4), είναι για το μπλέ χρώμα των RGB LED, φαίνονται με το μπλέ χρώμα στην Εικόνα 62.

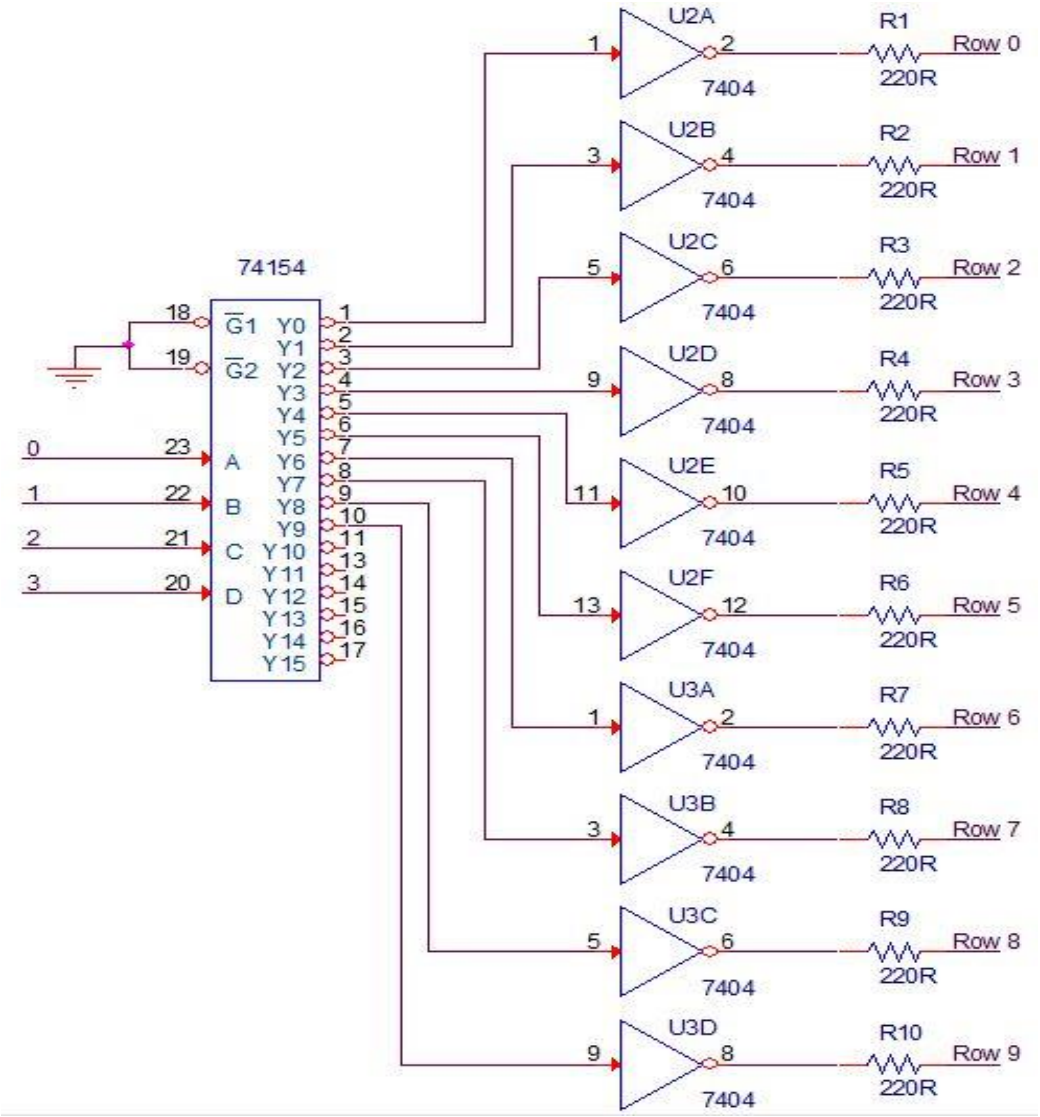
12. Παρόλα αυτά οι 25 συνδέσεις είναι πολλές γιατί έχουμε μόνο 40 pin στον μικροελεγκτή μας που από αυτά χρησιμοποιούνται τα περισσότερα για άλλες συνδέσεις όπως, SPI σύνδεση, κουμπιά επίτρεψης λειτουργιών από τον ESP8266 και η LCD οθόνη. Επομένως για να μειώσουμε κι άλλο τα pins που συνδέονται στον μικροελεγκτή μας (MSP-EXP432P401R) από 25 σε 13, έγινε η χρήση αποκωδικοποιητών/αποπλεκτών ως εξής:
    a. 4 pins για τον έλεγχο των γραμμών του πλέγματος των RGB LED μέσω του αποκωδικοποιητή/αποπλέκτη 4 σε 16 με την ονομασία 74154.
    b. 3 pins για τον έλεγχο των κόκκινο στηλών του πλέγματος των RGB LED μέσω του αποκωδικοποιητή/αποπλέκτη 3 σε 8 με την ονομασία 74155.
    c. 3 pins για τον έλεγχο των πράσινων στηλών του πλέγματος των RGB LED μέσω ενός δεύτερου 74155.
    d. 3 pins για τον έλεγχο των μπλέ στηλών του πλέγματος των RGB LED μέσω ενός τρίτου 74155.
13. Έτσι λοιπόν έγινε η δημιουργία της πλακέτας των αποκωδικοποιητών/αποπλεκτών, που περιέχει ένα 74154, 3x71455 ένα για κάθε χρώμα, επίσης 2 τσιπ 7404 πύλες NOT για την αναστροφή των εξόδων του 74154 για τον λόγο ότι τα LED μας είναι κοινής ανόδου και το 74154 μας δίνει LOW στην επιλογή αντί για HIGH που θέλουμε, καθώς και 10 αντιστάσεις 220Ω για την κάθε γραμμή που τροφοδοτείται. Περιέχει επίσης ένα 10KΩ ποτενσιόμετρο και ακόμα μία αντίσταση 220Ω για την σύνδεση της LCD οθόνης (βλέπε Breadboard 3). Τέλος περιέχει και 3 dip switches 10 διακοπτών το καθένα για την σύνδεση/αποσύνδεση όλων των pins που έχουν συνδεθεί πάνω στους μικροελεγκτές MSP-EXP432P401R και ESP8266, αποσυνδέοντας όλα τα pins

μας δίνεται η δυνατότητα επαναπρογραμματισμού των μικροελεγκτών (παράδειγμα περίπτωση αλλαγής πίνακα ονομασιών των ραφιών/συρταριών).



Εικόνα 63. Πλακέτα αποκωδικοποιητών/αποπλεκτών

14. Η πλακέτα αποκωδικοποιητών/αποπλεκτών είναι το μέσω διασύνδεσης του MSP-EXP432P401R με το πλέγμα των RGB LED ακολουθούν φωτογραφίες των αποκωδικοποιητών/αποπλεκτών και οι πίνακες αληθείας που τους διέπουν (Στα pins A,B,C,D (0,1,2,3) του αποκωδικοποιητή/αποπλέκτη 74154 και τα A,B,C (0,1,2) των 74155 (RGB) εννοούν το 2^0, 2^1, 2^2, 2^3 και αντίστοιχα 2^0, 2^1, 2^2 , ουσιαστικά μας υποδεικνύουν ποιό pin των αποκωδικοποιητών/αποπλεκτών είναι το LSB και ποιό το MSB).

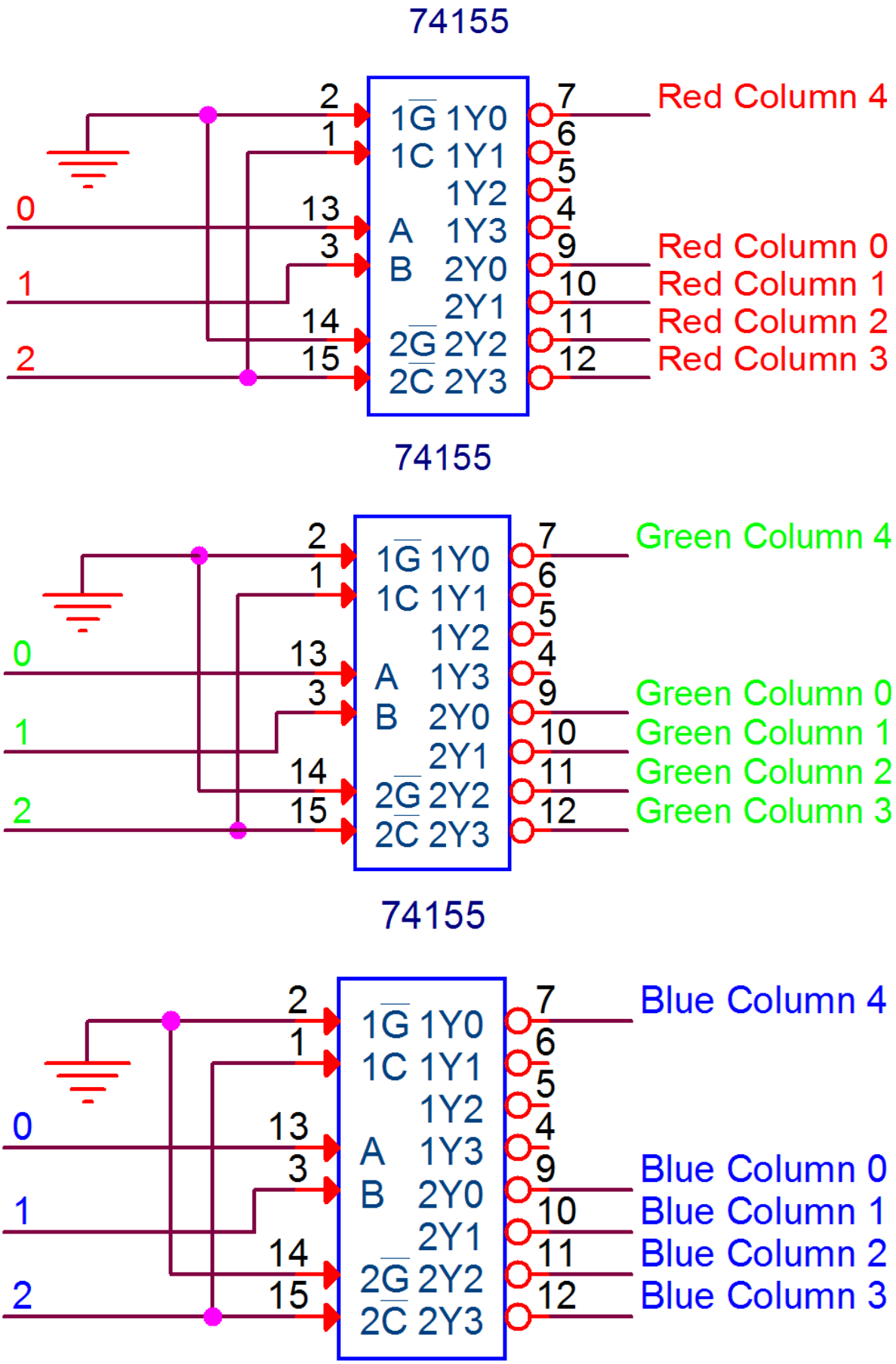


TRUTH TABLE

INPUTS						SELECTED OUTPUT (L)
$\overline{G1}$	$\overline{G2}$	D	C	B	A	
L	L	L	L	L	L	$\overline{Y0}$
L	L	L	L	L	H	$\overline{Y1}$
L	L	L	L	H	L	$\overline{Y2}$
L	L	L	L	H	H	$\overline{Y3}$
L	L	L	H	L	L	$\overline{Y4}$
L	L	L	H	L	H	$\overline{Y5}$
L	L	L	H	H	L	$\overline{Y6}$
L	L	L	H	H	H	$\overline{Y7}$
L	L	H	L	L	L	$\overline{Y8}$
L	L	H	L	L	H	$\overline{Y9}$
L	L	H	L	H	L	$\overline{Y10}$
L	L	H	L	H	H	$\overline{Y11}$
L	L	H	H	L	L	$\overline{Y12}$
L	L	H	H	L	H	$\overline{Y13}$
L	L	H	H	H	L	$\overline{Y14}$
L	L	H	H	H	H	$\overline{Y15}$
X	H	X	X	X	X	NONE
H	X	X	X	X	X	NONE

X : Don't Care

Εικόνα 64. Πίνακας αληθείας του αποκωδικοποιητή/αποπλέκτη 74154 - 4 σε 16


74155
2
1
13
3
14
15
0
1
2
1G
1C
A
B
2G
2C
1Y0
1Y1
1Y2
1Y3
2Y0
2Y1
2Y2
2Y3
7
6
5
4
9
10
11
12
Red Column 4
Red Column 0
Red Column 1
Red Column 2
Red Column 3
74155
Green Column 4
Green Column 0
Green Column 1
Green Column 2
Green Column 3
74155
Blue Column 4
Blue Column 0
Blue Column 1
Blue Column 2
Blue Column 3

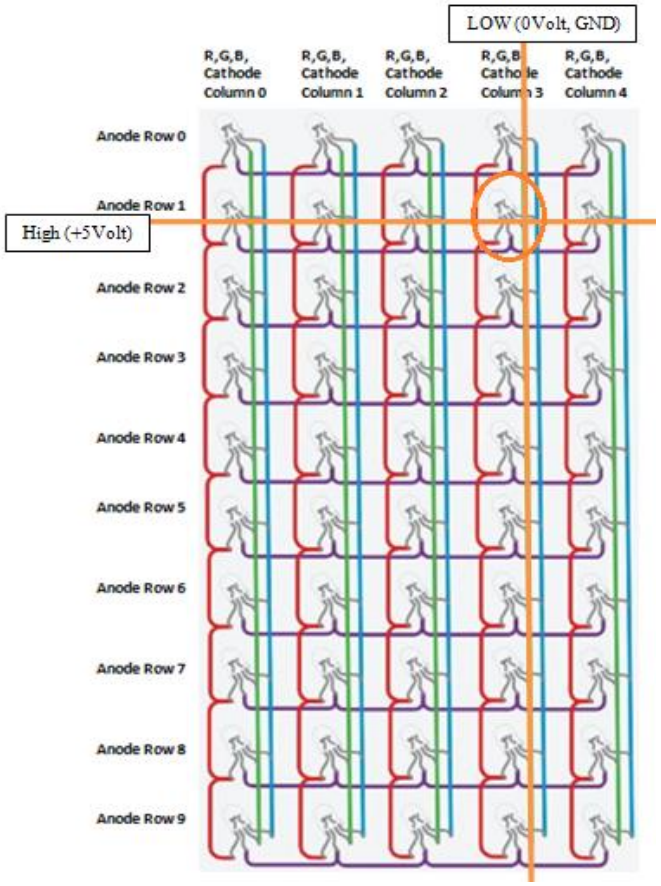
Inputs				Outputs							
Select			Strobe Or Data	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
C	B	A	G	2Y0	2Y1	2Y2	2Y3	1Y0	1Y1	1Y2	1Y3
X	X	X	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H
L	H	H	L	H	H	H	L	H	H	H	H
H	L	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H
H	H	L	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	H	H	L

Εικόνα 65. Πίνακας αληθείας του αποκωδικοποιητή/αποπλέκτη 74155 - 3 σε 8

15. Επομένως έχοντας το πλέγμα και τους αποκωδικοποιητές/αποπλέκτες μπορούμε δίνοντας έναν δεκαδικό αριθμό να τον μετατρέπουμε σε συντεταγμένες γραμμών και στηλών και να τροφοδοτούμε κατάλληλα τους αποκωδικοποιητές/αποπλέκτες για να μας ανάψουν το ζητούμενο LED. Η τροφοδοσία των αποκωδικοποιητών/αποπλεκτών γίνεται μέσω των δυαδικών εισόδων που είδαμε και στους πίνακες αληθείας A,B,C,D για τον 74154 και A,B,C για τον 74155, άρα με την μετατροπή του δεκαδικού αριθμού σε συντεταγμένες παίρνουμε τον αριθμό γραμμής και στήλης στο δεκαδικό σύστημα μετατρέποντάς το σε δυαδικό γνωρίζουμε πια A,B,C,D θα τροφοδοτήσουμε για να ανάψουμε το ζητούμενο LED.
16. Η εύρεση γραμμών γίνεται με το πηλίκο της ευκλείδειας διαίρεσης και η εύρεση στηλών γίνεται με το υπόλοιπο της (modulus), ο διαιρέτης είναι πάντα το 5 για τον λόγο ότι το κάθε κανάλι περιέχει 5 LED, οι τιμές των γραμμών είναι από 0 έως 9 και των στηλών από 0 έως 4. Παράδειγμα εύρεσης του RGB LED με το νούμερο 9:
    a. Πρώτα γίνεται ευκλείδεια διαίρεση και παίρνουμε το υπόλοιπο για τον έλεγχο γραμμής, $Γραμμή\ (Row) = \frac{(Ζητούμενος\ αριθμός - 1)}{5} = \frac{(9-1)}{5} = 1$ (το -1 γίνεται αυτόματα με τούς πίνακες μέσα στον κώδικα της ραφιέρας), άρα τροφοδοτούμε δυαδικά τον 74154 που ελέγχει τις γραμμές και επειδή είναι 4 bit, ο αριθμός που θα τροφοδοτήσουμε τις εισόδους D(MSB),C,B, A(LSB) για την εύρεση της γραμμής νούμερο 1 είναι: **0001**.
    b. Έπειτα παίρνουμε το υπόλοιπο της ευκλείδειας διαίρεσης για τον έλεγχο στήλης,

$Στήλη\ (Column) = (Ζητούμενος\ αριθμός - 1)\ \%\ 5 = (9-1)\ \%\ 5 = 3,$

άρα τροφοδοτούμε δυαδικά τον κατάλληλο 74155 που ελέγχει τις στήλες Red, ή Green, ή Blue αναλόγως την ποσότητα υλικών που έχει το κάθε ράφι και επειδή είναι 3 bit, ο αριθμός που θα τροφοδοτήσουμε τις εισόδους C(MSB),B,A(LSB) για την εύρεση της στήλης νούμερο 3 είναι: **011**. Ακολουθούν φωτογραφίες απόδειξης του παραδείγματος.



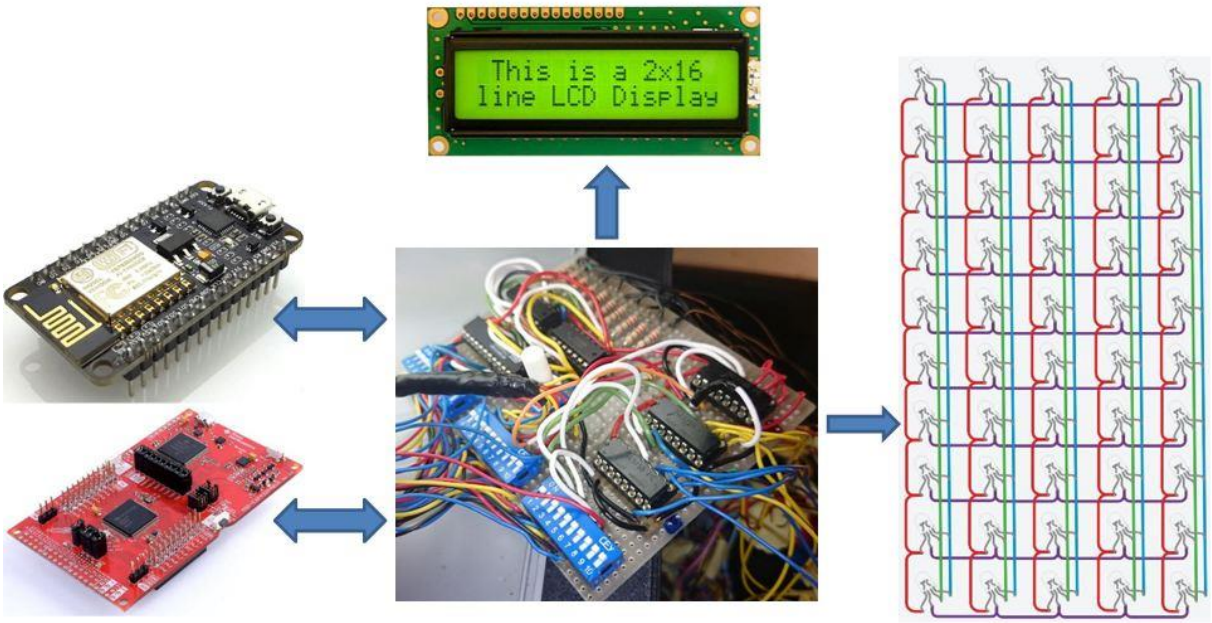
17. Στην συνέχεια γίνανε όλες οι συνδέσεις μεταξύ των 2 μικροελεγκτών, της πλακέτας των αποκωδικοποιητών/αποπλεκτών, του πλέγματος των RGB LED και της LCD οθόνης και όλα μαζί τα βάλαμε μέσα σε ένα προστατευτικό κουτί πάνω στην ραφιέρα.

Ακολουθούν οι πίνακες με τα pins όλων των συνδέσεων που γίνανε και οι φωτογραφίες του μπλοκ διαγράμματος των συνδέσεων, καθώς και του προστατευτικού κουτιού των υλικών.

Πίνακας 37. Πίνακας συνδέσεων του MSP-EXP432P401R

A/A	Ονομασία θύρας [Port Name (Px.n)]	Νούμερο Pin από τις θύρες συνδέσεων (J1 - J4)	Νούμερο Deep Switch (x) και νούμερο διακόπτη (n) (DSx(1-3).n(1-10)	Είσοδοι/Εξοδοι (Inputs/Outputs)	Περιγραφή
1	5V	21	DS1.1		VCC
2	GND	20,22	DS1.2		GND
3	P1.6	15	DS1.3	Input	MOSI
4	P1.7	14	DS1.4	Output	MISO
5	P1.5	7	DS1.5	Input	SPI CLK
6	P4.2	25	DS1.6	Output	74154 A(2^0)
7	P4.4	26	DS1.7	Output	74154 B(2^1)
8	P4.5	27	DS1.8	Output	74154 C(2^2)
9	P4.7	28	DS1.9	Output	74154 D(2^3)
10	P2.7	40	DS1.10	Input	General Check
11	P2.6	39	DS2.1	Output	LCD d7
12	P2.4	38	DS2.2	Output	LCD d6
13	P5.6	37	DS2.3	Output	LCD d5
14	P6.6	36	DS2.4	Output	LCD d4
15	P6.7	35	DS2.5	Output	LCD enable
16	P2.3	34	DS2.6	Output	LCD rs
17	P6.1	23	DS2.7	Output	Red 74155 A(2^0)
18	P5.1	33	DS2.8	Output	Red 74155 B(2^1)
19	P3.5	32	DS2.9	Output	Red 74155 C(2^2)
20	P4.6	8	DS2.10	Input	Change Material Quantity
21	P3.7	31	DS3.1	Output	Green 74155 A(2^0)
22	P2.5	19	DS3.2	Output	Green 74155 B(2^1)
23	P3.0	18	DS3.3	Output	Green 74155 C(2^2)
24	P5.0	13	DS3.4	Output	Blue 74155 A(2^0)
25	P5.2	12	DS3.5	Output	Blue 74155 B(2^1)
26	P3.6	11	DS3.6	Output	Blue 74155 C(2^2)
27	P4.1	5	DS3.7	Input	Excel

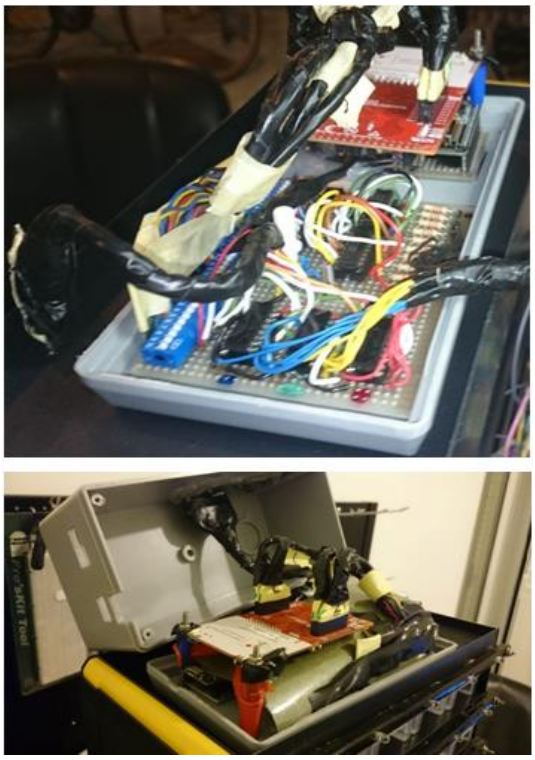
Πίνακας 38. Πίνακας συνδέσεων του ESP8266					
A/A	**Ονομασία θύρας Dx**	**Ονομασία Pin από τις θύρες συνδέσεων**	**Νούμερο Deep Switch (x) και νούμερο διακόπτη (n) (DSx(1-3).n(1-10)**	**Είσοδοι/Εξοδοι (Inputs/Outputs)**	**Περιγραφή**
1	GND	GND	DS1.2		GND
2	D7	MOSI	DS1.3	Output	MOSI
3	D6	MISO	DS1.4	Input	MISO
4	D5	SPI CLK	DS1.5	Output	SPI CLK
5	D3	GPIO0	DS1.10	Output	General Check
6	D1	GPIO5	DS2.10	Output	Change Material Quantity
7	D2	GPIO4	DS3.7	Output	Excel



Εικόνα 66. Μπλοκ διάγραμμα συνδέσεων των 2 μικροελεγκτών, με την πλακέτα των αποκωδικοποιητών/αποπλεκτών, την LCD οθόνη και το πλέγμα των RGB LED

Εικόνα 67. Πρόσοψη προστατευτικού κουτιού των 2 μικροελεγκτών, της πλακέτας των αποκωδικοποιητών/αποπλεκτών και της LCD οθόνης

Εικόνα 68. Πλάγια και πίσω όψη του προστατευτικού κουτιού των υλικών



Εικόνα 69. Εσωτερικά του προστατευτικού κουτιού

Εικόνα 70. Έξυπνη Ραφιέρα

Συμπεράσματα

Συνοψίζοντας, ο μικροελεγκτής MSP432P401R και το αναπτυξιακό του MSP-EXP432P401R επιλέχθηκε για τον λόγο της εξαιρετικά χαμηλής κατανάλωσης ισχύος μόλις 80μA/MHz στα 3.3 V, με συνολική κατανάλωση ρεύματος στα 48 MHz (μέγιστα Hz) μόνο 3.84 mA που τον καθιστά ιδανικό για φορητές εφαρμογές (Wearable applications), καθώς διαθέτει υψηλής απόδοσης επεξεργαστή ARM 32-Bit Cortex-M4F με FPU και MPU. Επίσης άλλος ένας λόγος επιλογής του είναι ότι διαθέτει ADC ανάλυσης 14 bits και ρυθμό δειγματοληψίας στο 1 MSPS που τον καθιστά ιδανικό για το διάβασμα ευαίσθητων αισθητήρων π.χ. πυρανόμετρο. Τέλος διαθέτει το εργαλείο Energy Trace που μας δίνει την δυνατότητα ένδειξης της κατανάλωσης του μικροελεγκτή και των περιφερειακών του σε πραγματικό χρόνο.

Μελλοντική επέκταση του οδηγού χρήσης του μικροελεγκτή MSP432P401R

Σε μελλοντική επέκταση της παρούσας πτυχιακής εργασίας στο κομμάτι του οδηγού χρήσης του MSP432P401R, θα μπορούσε να προστεθεί:

- το κομμάτι του διαχειριστή ελέγχου ισχύος [Power Control Manager (PCM)] που διαθέτει των έλεγχο λειτουργιών εξαιρετικά χαμηλής κατανάλωσης [Low Power Modes (LPM)] καθώς λειτουργίες ύπνου (Sleep) και βαθύ ύπνου (Deep Sleep).
- αναλυτικά το σύστημα ελέγχου του ρολογιού του συστήματος [Clock System (CS)]
- τον ελεγκτή χαρτογράφησης των θυρών [Port Mapping Controller (PMAP)]

και παρά πολλές ακόμα λειτουργιές που περιγράφονται στο Technical Reference Manual (slau356h.pdf) της Texas Instruments.

Μελλοντική βελτιστοποίηση της έξυπνης ραφιέρας

Σε μελλοντική βελτιστοποίηση της έξυπνης ραφιέρας θα μπορούσαμε να προσθέσουμε:

- Αυτόματη εύρεση ποσότητας υλικού στα ράφια, π.χ. μέσω δυναμοκυψέλων.

Βιβλιογραφία

Στην συγκεκριμένη πτυχιακή εργασία κυρίως χρησιμοποιήθηκε το διαδίκτυο (Internet) για την εύρεση πληροφοριών.

Ηλεκτρονικοί σύνδεσμοι (Web Links) ανά κεφάλαιο:

1ο Κεφάλαιο. Ιστορική αναδρομή των μικροελεγκτών

[1] http://www.circuitstoday.com/microcontroller-invention-history

[2] https://academics.uccs.edu/mlaubhan/ece3430/chiptimeline

[3] https://www.telegraph.co.uk/finance/newsbysector/epic/arm/8243162/History-of-ARM-from-Acorn-to-Apple.html

[4] http://www.vintagecalculators.com/html/the_calculator-on-a-chip.html

[5] https://files.vogel.de/vogelonline/vogelonline/files/8651.pdf

[6] https://en.wikichip.org/wiki/ti/tms1000

[7] https://en.wikipedia.org/wiki/AVR_microcontrollers

[8] https://en.wikipedia.org/wiki/TI_MSP432

[9] http://www.ti.com/corp/docs/company/history.html

2ο Κεφάλαιο. Αρχιτεκτονική μικροελεγκτή MSP432P4xx

[10] http://paulkilloran.com/arm/Lecture_8.pdf

[11] https://training.ti.com/msp432-mcus-and-arm-cortex-m4f-core

[12] https://en.wikipedia.org/wiki/ARM_Cortex-M

[13] https://www.ti.com/lit/ug/slau356h/slau356h.pdf

3ο Κεφάλαιο. Περιγραφή του υλικού (Hardware description)

[14] http://www.ti.com/lit/ug/slau597f/slau597f.pdf

[15] http://www.ti.com/lit/an/slaa656d/slaa656d.pdf

[16] http://www.ti.com/lit/ds/symlink/msp432p401r.pdf

4ο Κεφάλαιο. Εφαρμογές του MSP432P401R και σύγκριση του με άλλους μικροελεγκτές

Ηλεκτρονικός σύνδεσμος για τις εφαρμογές

[17] http://www.ti.com/lit/sg/slab034ad/slab034ad.pdf

Ηλεκτρονικοί σύνδεσμοι για την σύγκριση

[18] http://www.ti.com/lit/ml/swrp156/swrp156.pdf

[19] https://store.arduino.cc/usa/arduino-uno-rev3

[20] http://www.ti.com/lit/ds/symlink/msp432p401r.pdf

[21] http://www.ti.com/tool/MSP-EXP432P401R

5ο Κεφάλαιο. Οδηγός χρήσης του Code Composer Studio (CCS) IDE και ανάλυση βασικών καταχωρητών (Registers)

[22] https://www.ti.com/lit/ug/slau356h/slau356h.pdf

[23] http://www.ti.com/lit/ug/slau157ar/slau157ar.pdf

6ο Κεφάλαιο. Οδηγός χρήσης του Energia IDE και ανάλυση βασικών εντολών του

[24] http://energia.nu/

[25] http://energia.nu/reference/

[26] https://www.tinkercad.com/#/

7ο Κεφάλαιο. Υλοποίηση εφαρμογής ταξινόμησης υλικών σε ράφια και κατασκευή της ραφιέρας

Ηλεκτρονικοί σύνδεσμοι για τον μικροελεγκτή Arduino ESP8266

[27] https://www.arduino.cc/

[28] https://www.arduino.cc/en/Reference/SPI

[29] https://arduino-esp8266.readthedocs.io/en/latest/

Ηλεκτρονικοί σύνδεσμοι για το Blynk

[30] https://www.blynk.cc/

[31] https://www.blynk.cc/getting-started/

[32] http://docs.blynk.cc/#getting-started-getting-started-with-the-blynk-app-4-auth-token

[33] https://play.google.com/store/apps/details?id=cc.blynk

Ηλεκτρονικός σύνδεσμος για το PLX-DAQ-v2.11

[34] http://forum.arduino.cc/index.php?topic=437398.msg3251256#msg3251256

Παραρτήματα

ΠΑΡΑΡΤΗΜΑ Α - ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ ΤΑΞΙΝΟΜΗΣΗΣ ΥΛΙΚΩΝ ΣΕ ΡΑΦΙΑ

Τα αρχεία της εφαρμογής ταξινόμησης υλικών σε ράφια για την έξυπνη ραφιέρα Master_ESP8266_Rafiera.ino , Slave_msp432_Rafiera.ino και το αρχείο καταγραφής της ραφιέρας στο Microsoft Excel με την ονομασία PLX-DAQ-v2.11 μπορούν να ζητηθούν από το εργαστήριο Ηλεκτρονικών του τμήματος Ηλεκτρολόγων Μηχανικών Τ.Ε. της Σχολής Τεχνολογικών Εφαρμογών του ΤΕΙ Κρήτης ή να αποστείλετε προσωπικό μήνυμα στο e-mail **john_kalogerakis_12@hotmail.com** για να σας αποσταλούν τα αρχεία της εφαρμογής.