ΚΕΦΑΛΑΙΟ 1

Αναπτυξιακά Εργαλεία για Μικροελεγκτές 8-Bit AVR Microchip

1.1. Εισαγωγή

Στο μέρος αυτού του βιβλίου θα εξετάσουμε και θα εφαρμόσουμε στην πράξη την χρήση των μικροελεγκτών AVR. Για τον σκοπό αυτό θα χρησιμοποιήσουμε διάφορες διατάξεις με κυκλώματα και συνδέσεις. Ωστόσο για να έχουμε το επιθυμητό αποτέλεσμα απαιτείται να έχουμε στην κατοχή μας έτοιμο προς χρήση τον κατάλληλο συνδυασμό υλικών και κυκλωμάτων (hardware), αλλά και των αντίστοιχων προγραμμάτων, οδηγών, εφαρμογών, αυτό που ονομάζουμε λογισμικό μέρος (software). Πριν ξεκινήσουμε οποιαδήποτε κατασκευή, ή εφαρμογή λογισμικού θα πρέπει να αναφέρουμε τι θα χρειαστούμε για την υλοποίηση αυτών των κατασκευών, αλλά και τον τρόπο με τον οποίο θα τις υλοποιήσουμε.

Όσοι ασχολείστε για πρώτη φορά με μικροελεγκτές πιθανότατα θα έχετε πληθώρα ερωτήσεων, όπως το τί είναι ένα αναπτυξιακό εργαλείο και σε τι χρησιμεύει, ή τί είναι ένας προγραμματιστής (Programmer),η ένας συμβολομεταφραστής, πως λειτουργούν, την σειρά και τα βήματα που θα πρέπει να ακολουθήσει κάνεις για την επιτυχή ανάπτυξη ενός συστήματος κλπ.

Στα παρακάτω κεφάλαια δίνονται οι απαντήσεις στην πληθώρα των ερωτήσεων που έχουνε κατά νου οι αναγνώστες του βιβλίου. Έτσι για να έχουμε σωστή σειρά στο πρώτο μέρος αυτού του κεφαλαίου θα ασχοληθούμε με τα αναπτυξιακά εργαλεία των μικροελεγκτών, λογισμικό (Software) και υλικό (hardware), ξεκινώντας πρώτα με τα εργαλεία λογισμικού.

Μέχρι τώρα εξετάσαμε την λειτουργία και την διάταξη των διαφόρων εσωτερικών μονάδων ενός μικροελεγκτή AVR ATmega αλλά και τους τρόπους με τους οποίους μπορεί να συνδεθεί ένας μικροελεγκτής με το εξωτερικό περιβάλλον.

Ήρθε η ώρα να αναλύσουμε αλλά και να εξετάσουμε στην πράξη τα εργαλεία με τα οποία θα μπορέσουμε να σχεδιάσουμε,να κατασκευάσουμε και να αναπτύξουμε το κατάλληλο λογισμικό ώστε το σύστημα μας να λειτουργεί πραγματικά για της εργασίες που θέλουμε να πραγματοποιήσουμε.

Όπως αντιλαμβάνεστε δυο είναι οι ομάδες εργαλείων που θα εξετάσουμε στο παρακάτω κεφάλαιο.

1. **Εργαλεία Λογισμικού**

   Όπως γνωρίζουμε τα εργαλεία λογισμικού είναι προγράμματα με τα οποία μπορούμε να δημιουργήσουμε κώδικα για εφαρμογές,να πραγματοποιήσουμε αποσφαλμάτωση,έλεγχο της εφαρμογής,εξαγωγή εκτελέσιμου αρχείου, σχεδίαση πλακέτας PCB,προσομοίωση, παρακολούθηση έργου κ.λπ.

2. **Εργαλεία Υλικού**

   Για τα εργαλεία του υλικού μέρους θα αναφερθούμε εκτενώς στο κεφάλαιο αναπτυξιακές κάρτες και εργαλεία.

Θα ξεκινήσουμε πρώτα από τα εργαλεία του λογισμικού μέρους.

1.2. Συμβολομεταφραστής Προγραμμάτων

Είναι αναγκαίο πριν προχωρήσουμε βαθύτερα στην ανάπτυξη εφαρμογών και κατασκευών να αναλύσουμε λίγο περισσότερο το περιβάλλον με το οποίο θα εργαστούμε και θα δημιουργήσουμε το υλικολογισμικό μας.

Είναι φυσικό όταν για πρώτη φορά ακούσουμε την λέξη συμβολομεταφραστής συνήθως το πρώτο άκουσμα είναι ξενικό (Assembler),μας γεννάται το ερώτημα τι ακριβώς είναι αυτό το εργαλείο λογισμικού.

Τι είναι λοιπόν ένας συμβολομεταφραστής;

Σε απόλυτη απάντηση ένας συμβολομεταφραστής ουσιαστικά είναι μια εφαρμογή οι οποία μεταφράζει τον πηγαίο κώδικα (source code) σε αντικειμενικό κώδικα (object code),δημιουργώντας ταυτόχρονα το αρχείο προγραμματισμού (Hex).

Για να το κάνουμε πιο κατανοητό, αναφέρουμε ένα παράδειγμα με μια εντολή συμβολικής γλώσσας.

```
SEI:
```

Ορίζει τη σημαία γενικής διακοπής (I) στο SREG (Status Register) καταχωρητής κατάστασης.

1001	0100	0111	1000

Σχήμα 2.1. *Ο* ***16-Bits Opcode*** *που αντιστοιχεί στον λειτουργικό κώδικα.*

Τρόπος Λειτουργίας

Ο μικροελεγκτής διαβάζει την εντολή από την μνήμη προγράμματος και στην συνέχεια αποκωδικοποιεί την αντιστοιχεί δυαδική λέξη. Συγκεκριμένα, η εντολή **SEI** είναι συμβολική παράσταση (όπως όλες οι εντολές) και για τους μυημένους αντιστοιχεί στον λειτουργικό κώδικα **(Opcode $9478).**

Καθώς γνωρίζουμε οι μικροελεγκτές δεν αντιλαμβάνονται άμεσα τους χαρακτήρες των μνημονικών συμβόλων, **(SEI)** στο παράδειγμά μας αλλά τον αντίστοιχο λειτουργικό κώδικα των εντολών. Επομένως θα ήταν τεχνικά δύσκολο να χειριστούμε τους λειτουργικούς κώδικες. Για αυτό τον λόγο εφαρμόζουμε και χρησιμοποιούμε την τεχνική των μνημονικών συμβόλων καθώς είναι εύκολα αντιληπτή και διαχειρίσιμη.

Οπότε θα χρειαστούμε ένα κατάλληλο πρόγραμμα το οποίο θα μεταφράζει τα συμβολικά ονόματα σε λειτουργικούς κώδικες.

Η εφαρμογή για την εργασία αυτή ονομάζεται **συμβολομεταφραστής** ή **Assembler**.

Επίσης ο συμβολομεταφραστής εκτός από τον αντικειμενικό κώδικα δημιουργεί και μια άλλη μορφή αρχείου (Hex) το οποίο περιέχει τον κώδικα της εφαρμογής την οποία φορτώνουμε μέσω μιας κατάλληλης συσκευής προγραμματισμού (device programmer) στον μικροελεγκτή μας.

Τέλος ο συμβολομεταφραστής δημιουργεί και ένα αρχείο EEPROM εφόσον συντρέχουν βέβαια οι κατάλληλες προϋπόθεσης δηλαδή να έχουμε ορίσει στο αρχείο του κώδικα μας κάποια δεδομένα.

Σημείωση:

SEI = ($9478, και ο αντίστοιχος δυαδικός = b 1001 0100 0111 1000).

1.3. Προσομοίωση των Μικροελεγκτών AVR Microchip Simulator

Όσοι εργάζονται με τους μικροελεγκτές θα έχουν πολλές φορές βρεθεί στην θέση να φορτώνουν το πρόγραμμα τους αλλά αυτό να τους απογοητεύει, καθώς δεν ανταποκρίνεται στις προσδοκίες τους με αποτέλεσμα να μην γνωρίζουν τι ακριβώς πάει λάθος στο σύστημα.

- Ευθύνεται η εφαρμογή;
- Το υλικό μέρος αυτού του συστήματος;
- Κάποιος αισθητήρας;
- Η ακόμα και λάθος στην σχεδίαση;

Σε αυτές λοιπόν της καταστάσεις που είναι καθημερινότητα για όσους ασχολούνται με τους μικροελεγκτές και πρέπει να εντοπίσουν την πηγή του σφάλματος ώστε να προβούν στις απαραίτητες τροποποιήσεις, να επαναφορτώσουν τον νέο κώδικα αφού πρώτα τον μεταφράσουν ξανά και ούτω καθ' εξής. Αυτή η διαδικασία αποδεικνύεται κοπιαστική και χρονοβόρα.

Αντί λοιπόν να μεταφέρουμε κάθε λίγο τον κώδικα στον μικροελεγκτή για να προβούμε στην αποσφαλμάτωση (debug) του προγράμματος υπάρχει η λύση και αυτή ονομάζεται:

Προσομοιωτής προγραμμάτων:

Προσομοιώνει τους μικροελεγκτές στον υπολογιστή μας χωρίς να χρειαστεί να μεταφέρουμε τον λειτουργικό κώδικα μηχανής στον μικροελεγκτή μας.

Το πρόγραμμα αυτό με το οποίο μπορούμε να κάνουμε προσομοίωση λογισμικού σε ένα μικροελεγκτή ονομάζεται προσομοιωτής (simulator), μέσω του αυτού προγράμματος μας παρέχεται η δυνατότητα να εκτελούμε βήμα προς βήμα τον κώδικα της εφαρμογής μας παρατηρώντας ταυτόχρονα μέσα από το παραθυρικό περιβάλλον του υπολογιστή την κατάσταση του μικροελεγκτή, τα περιεχόμενα των (καταχωρητών εργασίας, την κατάσταση χρονιστών/μετρητών, τα περιεχόμενα low-high των θυρών Εισόδου-Εξόδου,κ.λπ.

Μέσα από την προσομοίωση έχουμε την ευχέρεια να εντοπίσουμε την πηγή του σφάλματος όταν υπάρξει, δίνοντας μας ταυτόχρονα την δυνατότητα να διορθώσουμε τον πηγαίο κώδικα και να εξασφαλίσουμε την σωστή λειτουργία του,όταν τελικά φορτωθεί στον μικροελεγκτή.

1.4. Οι συμβολομεταφραστές των Μικροελεγκτών AVR ATMEL-Microchip

Λόγω του ότι η συγγραφή του βιβλίου που κρατάτε στα χεριά σας άρχισε μετά την εξαγορά της ATMEL από την Microchip, οι συμβολομεταφραστές που θα αναφέρουμε παρακάτω αφορούν την οικογένεια μικροελεγκτών AVR ATMEL, Microchip. Για την ιστορία αναφέρουμε εκείνη την εποχή ο πιο διαδεδομένος και διαθέσιμος συμβολομεταφραστής ήταν αυτός που παρείχε η εταιρία ATMEL για τους μικροελεγκτές AVR με την ονομασία (Avrasm) ο ποιος κάλυπτε όλοι την σειρά των μικροελεγκτών AVR.

Σημείωση:

Η εκτέλεση του συμβολομεταφραστή Avrasm πραγματοποιούταν (πιθανώς να τον χρησιμοποιούν και στις μέρες μας) σε περιβάλλον MS DOS. Για το περιβάλλον Windows υπήρχε η έκδοση (wavrasm) και εκτελείτο σε win95, win98, win2000, αυτά όμως ανήκουν στο παρελθόν.

Σήμερα υπάρχουν πολλοί συμβολομεταφραστές για τους μικροελεγκτές AVR. Εμείς θα αναφερθούμε σε αυτούς που χρησιμοποιούνται από την πλειοψηφία της παγκόσμιας κοινότητας.

1.5. Ολοκληρωμένο Περιβάλλον AVR Microchip Studio 7

Το AVR Microchip Studio αποτελεί ένα ολοκληρωμένο αναπτυξιακό περιβάλλον με πολλά και πλούσια εργαλεία για όλους τους μικροελεγκτές AVR 8 η 32 Bit, όπως και για την σειρά μικροελεγκτών τύπου (SAM).

Το AVR Studio επιτρέπει την εκτέλεση προγραμμάτων με τον ενσωματωμένο εξομοιωτή των εντολών της γλωσσάς των μικροελεγκτών AVR. Έτσι μπορούμε να εκτελέσουμε προγράμματα σε επίπεδο πηγαίου κώδικα είτε είναι γραμμένος σε Assembly AVR η και σε C η C++. Το AVR Studio μας παρέχει

- ένα συντάκτη για την συγγραφή (editor) του κώδικα μας
- Τον συμβολομεταφραστή για την δημιουργία αντικειμενικού κώδικα, των αρχείων (hex),
- τον εξομοιωτή για τον έλεγχο βήμα-βήμα του πηγαίου κώδικα,και πολλά άλλα τα οποία θα συναντήσουμε στην συνέχεια καθώς θα χρησιμοποιούμε το AVR Studio για της διάφορες πρακτικές και θεωρητικές άσκησης είτε κάνοντας προσομοίωση προγράμματος η γράφοντας ένα πηγαίο κώδικα.

Το AVR Studio 7 συνδέεται επίσης απρόσκοπτα στα προγράμματα εντοπισμού σφαλμάτων, τους προγραμματιστές και της πλακέτες (Kit) ανάπτυξης που υποστηρίζουν τις συσκευές AVR και SAM.

Επιπλέον, το Studio 7 περιλαμβάνει το Atmel Gallery, ένα ηλεκτρονικό κατάστημα εφαρμογών που σας επιτρέπει να επεκτείνετε το αναπτυξιακό σας περιβάλλον με plug-ins που αναπτύσσονται από τη Microchip.

Καθώς και προμηθευτές λογισμικού και ενσωματωμένου λογισμικού τρίτων κατασκευαστών.

Το AVR Studio μπορεί να διασυνδεθεί με όλες της παλιές αναπτυξιακές κάρτες της ATMEL και τώρα με όλες της καινούργιες της Microchip, όπως και με αναπτυξιακές κάρτες τρίτων κατασκευαστών πχ Arduino και τους κλώνους,USB ASP κ.λπ.

Επίσης για το σκοπό αυτό υπάρχουν και άλλα προγράμματα όπως το WINAVR, και ο GCC Compiler,ο μεταγλωττιστής C της IAR κλπ.

Το AVR Studio μπορείτε να το κατεβάσετε μέσα από το Site της Microchip. Υπάρχουν δυο εκδόσεις εγκατάστασης για το Studio 7.0, μια μέσω διαδικτύου και μια χωρίς πρόσβαση στο διαδίκτυο (Internet).

1.5.1. Απαιτήσεις Συστήματος

Εγκατάσταση και Οδηγίες Εγκατάστασης.

Υποστηριζόμενα λειτουργικά συστήματα.

- Windows 7 Service Pack 1 ή νεότερη έκδοση.
- Windows Server 2008 R2 Service Pack 1 ή νεότερη έκδοση.
- Windows 8 / 8.1.
- Windows Server 2012 και Windows Server 2012 R2.
- Windows 10 / Windows 11.
- Υποστηριζόμενες αρχιτεκτονικές: 32-bit (x86) 64-bit (x64).

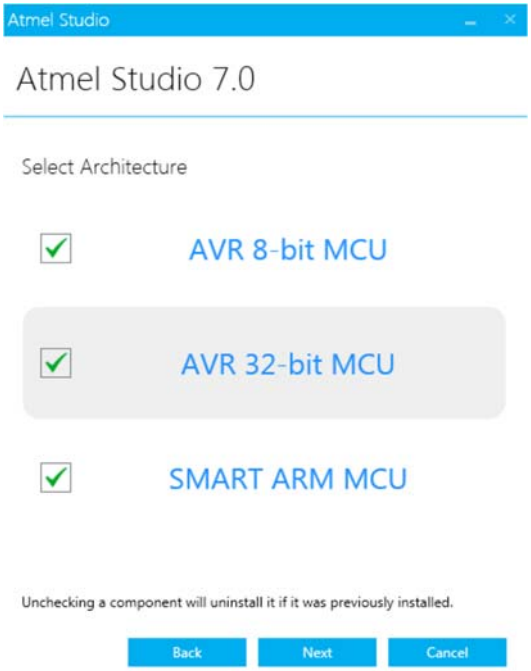
Απαιτήσεις υλικού.

- Υπολογιστής με επεξεργαστή 1,6 GHz ή ταχύτερο.
- 2 GB RAM για x86 - 2 GB RAM για x64.
- Μια πρόσθετη μνήμη RAM 512 MB εάν εκτελείται σε μια εικονική μηχανή
- Και τουλάχιστον 6 GB διαθέσιμου χώρου στο σκληρό δίσκο.

1.5.2. Εγκατάσταση προγράμματος AVR Microchip Studio 7.0

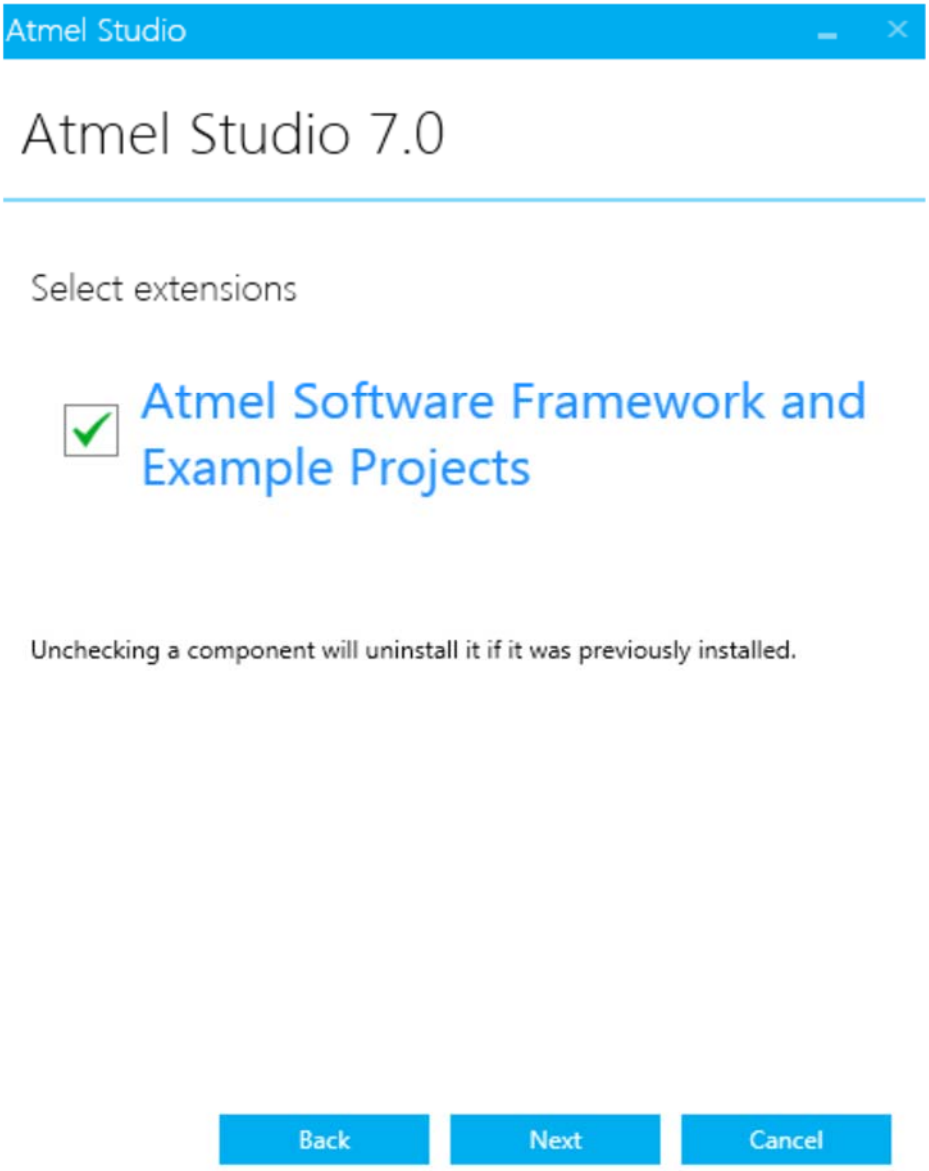
Εικόνα 2.1

Η πρώτη οθόνη που εμφανίζεται στην εικόνα **2.2** μας προτρέπει να επιλέξουμε την αρχιτεκτονική του μικροελεγκτή,να την αφήσουμε ως έχει η να απόεπιλέξουμε π.χ της επιλογές AVR 32 Bit και SMART ARM CPU,αν δεν θα εργαστούμε σε αυτού του τύπου μικροελεγκτές. Στο παράδειγμα μας τα αφήνουμε ως έχουν.



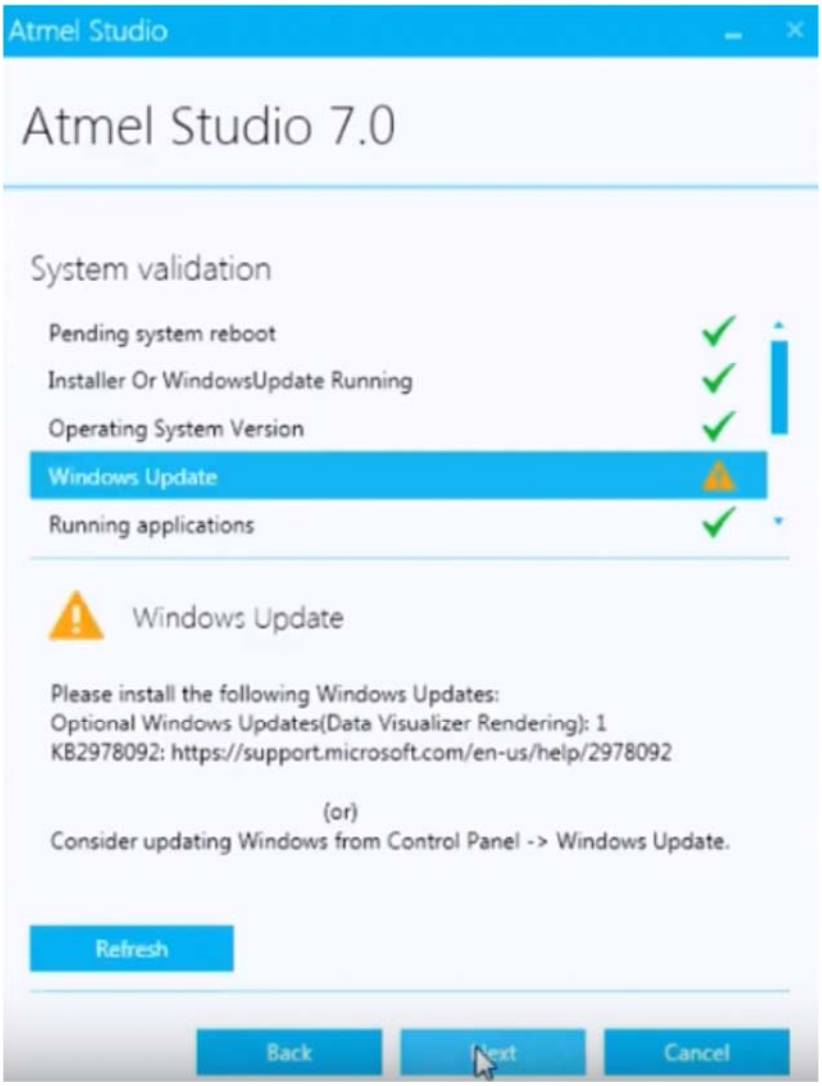
Εικόνα 2.2

Η αμέσως επόμενη οθόνη εικόνα 2.3 εμφανίζει την επιλογή για επέκταση του λογισμικού Framework της ATMEL, αυτή την επιλογή την προσπερνάμε και αν χρειαστεί δύναται να προχωρήσουμε αργότερα σε αναβάθμιση μέσα από το περιβάλλον του AVR Microchip - Studio 7.0.

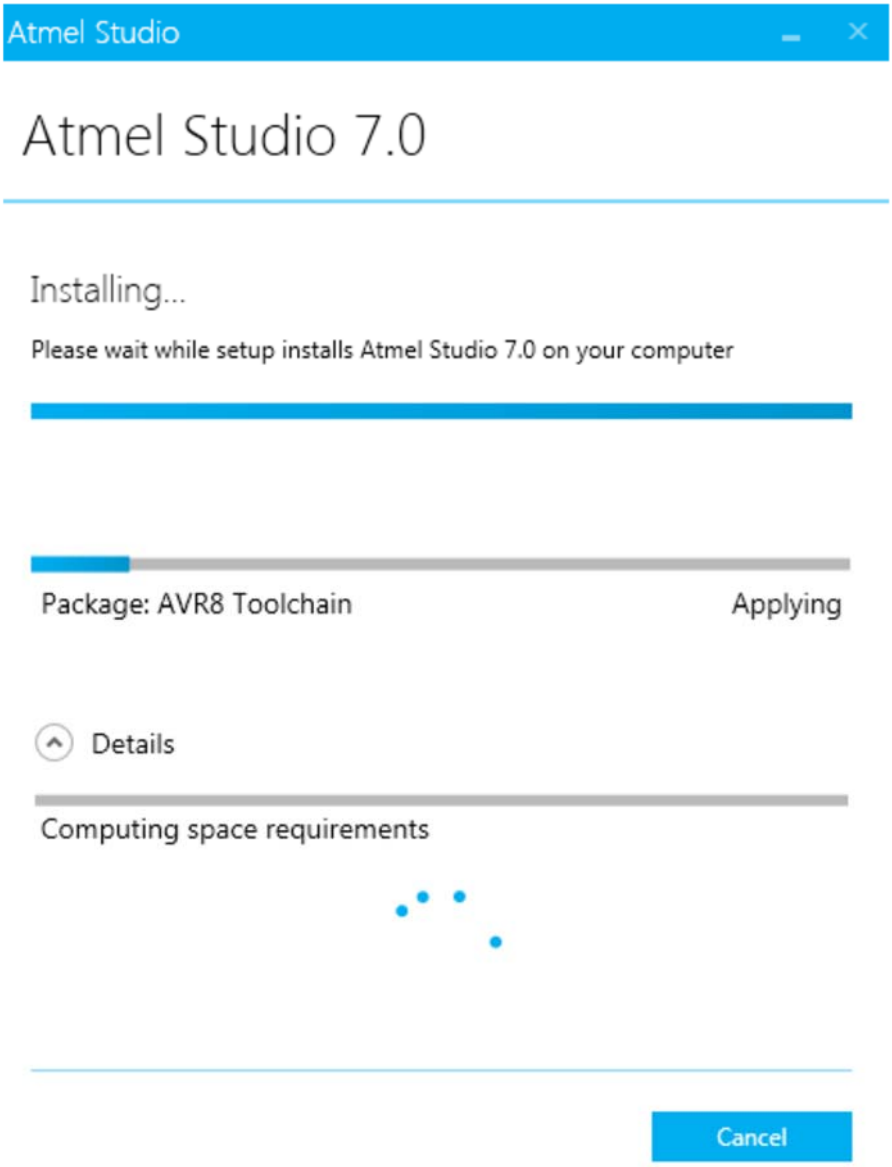


Εικόνα 2.3

Στην τελευταία οθόνη εγκατάστασης του προγράμματος εικόνα 2.4 γίνεται επικύρωση και έλεγχος συστήματος. Ενδεχομένως μπορεί να χρειαστείτε αναβάθμιση των Windows η κάποιας οδηγίας αυτών.



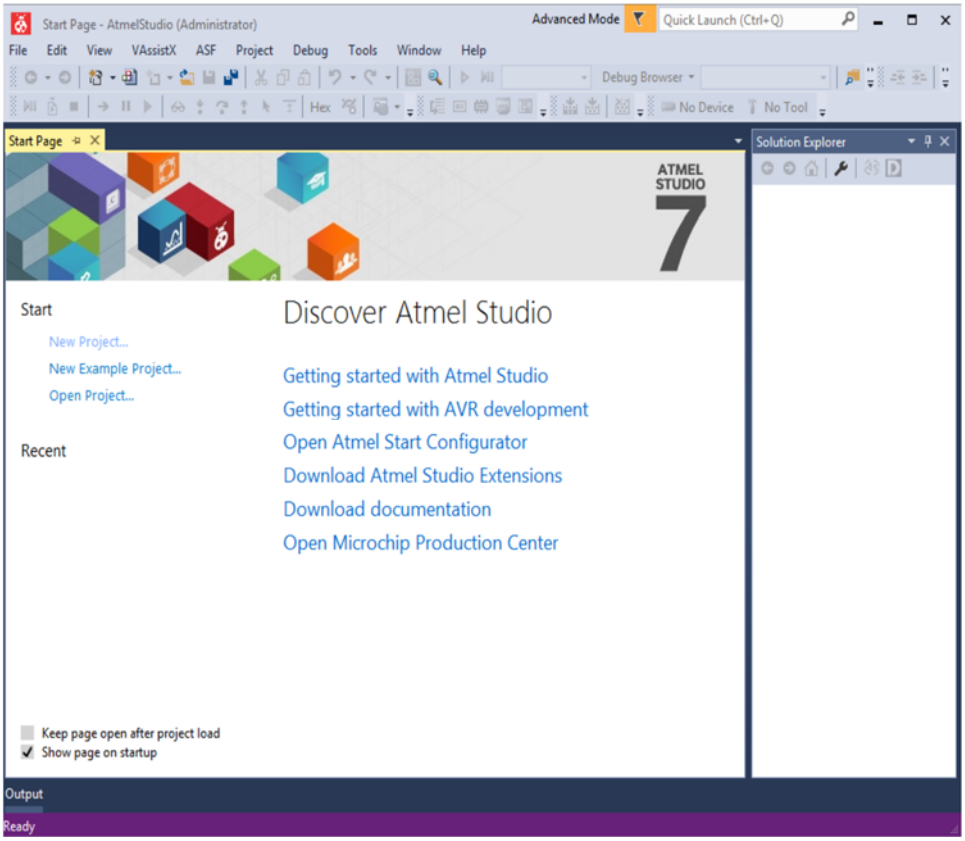
Εικόνα 2.4



Εικόνα 2.5. *Στο σημείο αυτό ξεκινά η εγκατάσταση του AVR STUDIO 7.*

Η διαδικασία αυτή θα χρειαστεί αρκετά λεπτά έως ότου ολοκληρωθεί η εγκατάσταση του STUDIO 7.

Τέλος στην επόμενη που θα εμφανισθεί εκτός απροόπτου έχουμε ολοκλήρωση την εγκατάσταση του STUDIO 7.



***Εικόνα 2.6.** Η αρχική εικόνα του περιβάλλοντος εργασίας του STUDIO 7.*
File - Edit - View - VAssistX - ASF - Project - Debug - Tools - Window – Help.

Όταν εκτελούμε το STUDIO 7 εμφανίζεται η παραπάνω αρχική οθόνη εργασίας η οποία είναι και η κεντρική οθόνη του STUDIO 7,οπου μέσω αυτής έχουμε τον πλήρη έλεγχο του προγράμματος για όλες της διαδικασίες και διασυνδέσεις που θα χρησιμοποιούμε για την ολοκλήρωση των εργασιών μας.

Σημείωση: Όταν εκτελούμε το ATMEL- Microchip STUDIO 7 και είναι διασυνδεμένο οποιοδήποτε υλικό που είναι συμβατό με το STUDIO 7 π.χ μια κάρτα ανάπτυξης (Development Board) η ένα προγραμματιστή (Programmer) αυτό αμέσως αναγνωρίζεται από το λογισμικό που είναι στην προκειμένη περίπτωση το (AVR-Microchip STUDIO 7). Εάν δεν αναγνωριστεί κανένα υλικό, δεν θα είναι το σωστό υλικό η δεν θα υπάρχουν οι ενδεδειγμένη οδηγοί (Drivers) κλπ.

Στα παρακάτω παραδείγματα χρησιμοποιούμε το πρόγραμμα προσομοίωσης του STUDIO7 για την διαδικασία εκτέλεσης του πηγαίου κώδικα.

1.5.3. AVR Microchip Studio 7

Τώρα που έχουμε εγκαταστήσει την εφαρμογή STUDIO 7 ας ξεκινήσουμε με της βασικές λειτουργίες αυτού του προγράμματος κατανοώντας ότι για ένα αρχάριο η για έναν που ασχολείται για πρώτη φορά με το ATMEL- Microchip STUDIO 7 θα φαίνεται αρκετά δύσκολα στην αρχή με τις τόσο αναλυτικές δυνατότητες και πολλές λειτουργίες που έχει το STUDIO 7.

Ωστόσο με την καθημερινή χρήση του αρχίζει και ξεδιαλύνει η ομίχλη γύρω από το STUDIO 7, κάνοντας μας να θέλουμε όλο και περισσότερο να χρησιμοποιήσουμε όλες της λειτουργίες και της εφαρμογές που μας παρέχει το AVR STUDIO 7.

Για τον λόγο αυτό θα προχωρήσουμε μαζί βήμα-βήμα τουλάχιστον στης βασικές εργασίες αναλύοντας την κάθε λειτουργία που θα χρησιμοποιήσουμε κατά την διάρκεια των μαθήματων και πειραμάτων αυτού του βιβλίου.

Η οθόνη έναρξης του Microchip Studio 7

Στην αρχική οθόνη και στην άνω περιοχή όπως παρατηρούμε στην **εικόνα 2.6** υπαρχή η γραμμή των λειτουργιών **File – Edit – View – VassistX – ASF – Project – Debug – Tools – Window – Help.**

Για να γράψουμε ένα πρόγραμμα θα πρέπει να μεταφέρουμε τον κέρσορα - δείκτη του ποντικιού μας στην επιλογή File Αρχείο κάνοντας κλικ.

Στην φόρμα που εμφανίζεται παρατηρούμε τις διάφορες δυνατότητες επιλογών.

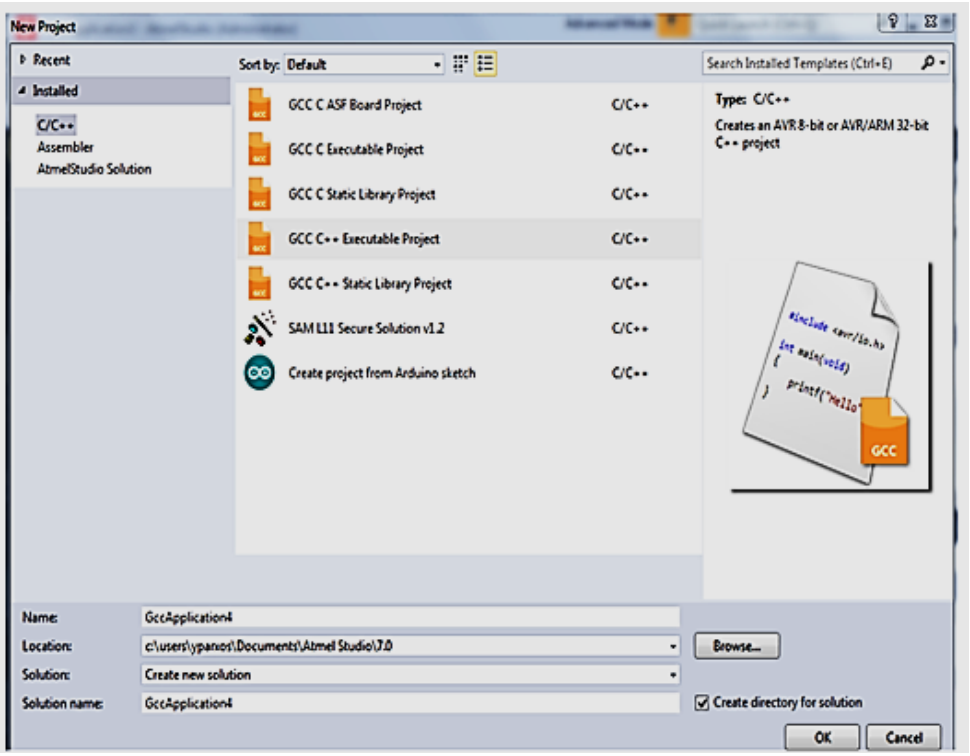
Προς το παρών επιλέγουμε **New → Project** και στην οθόνη που εμφανίζεται μας παρέχεται η δυνατότητα να επιλέξουμε την γλωσσά προγραμματισμού και τον μεταγλωττιστή που θα χρησιμοποιήσουμε.

Επιλέγουμε για αρχή **GCC C Executable Project**. Με αυτή την επιλογή δημιουργούμε αρχεία κώδικα σε γλώσσα προγραμματισμού C. Θα μπορούσαμε να επιλέξουμε κάτι διαφορετικό όπως π.χ Assembler και αμέσως με τα AVR Assembler Project για την δημιουργία κώδικα σε γλώσσα Assembly language.

Στο κάτω μέρος της οθόνης και στην επιλογή **Name** γράφουμε το όνομα του αρχείου όπως εμείς θέλουμε να ονομάζεται.

Στην επόμενη επιλογή επιλέγουμε την τοποθεσία αποθήκευσης του αρχείου μας μέσω της επιλογής (**Browse**) η το αφήνουμε ως έχει.

Εάν θέλουμε δημιουργούμε και κατάλογο για την λειτουργία Solution επιλέγουμε κλικ στο (check box) τετραγωνάκι που υπαρχή κάτω δεξιά στην οθόνη, **εικόνα 2.7**.



Εικόνα 2.7. *Επιλογή γλώσσας προγραμματισμού και μεταγλωττιστή.*

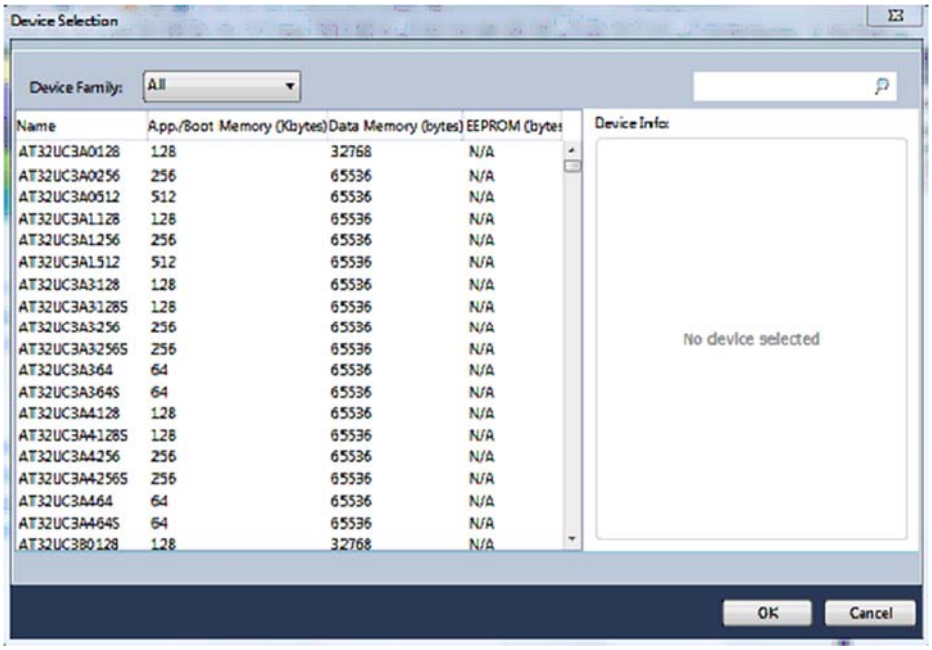
Επιλέγουμε **OK** και οδηγούμεθα στην επόμενη οθόνη.

Εδώ εμφανίζεται η δυνατότητα επιλογής του μικροελεγκτή που θα χρησιμοποιήσουμε για την εργασία μας.

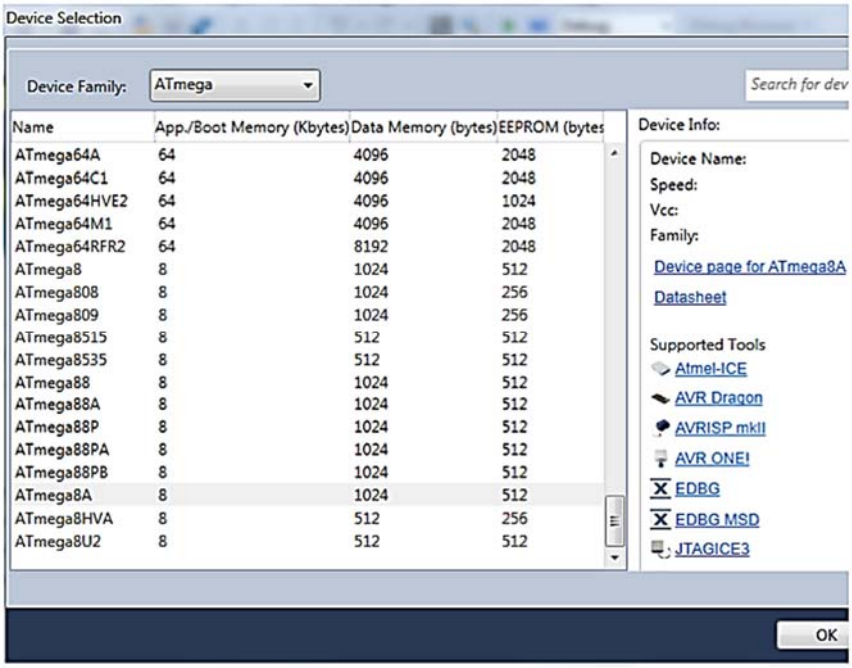
Όπως παρατηρούμε στην παρακάτω εικόνα η εφαρμογή μας εμφανίζει το σύνολο των μικροελεγκτών της ATMEL και της Microchip καθώς οι νέοι μικροελεγκτές κατασκευάζονται πλέον από την Microchip.

Στην επιλογή All μας δίδεται η δυνατότητα να επιλέξουμε την οικογένεια των μικροελεγκτών, καθώς εμφανίζονται όλοι οι τύποι των μικροελεγκτών της ATMEL - Microchip.

Το σύνολο των μικροελεγκτών της ATMEL και της Microchip. Στην οθόνη της εικόνας 2.8, επιλέγουμε την οικογένεια των μικροελεγκτών ATmega.



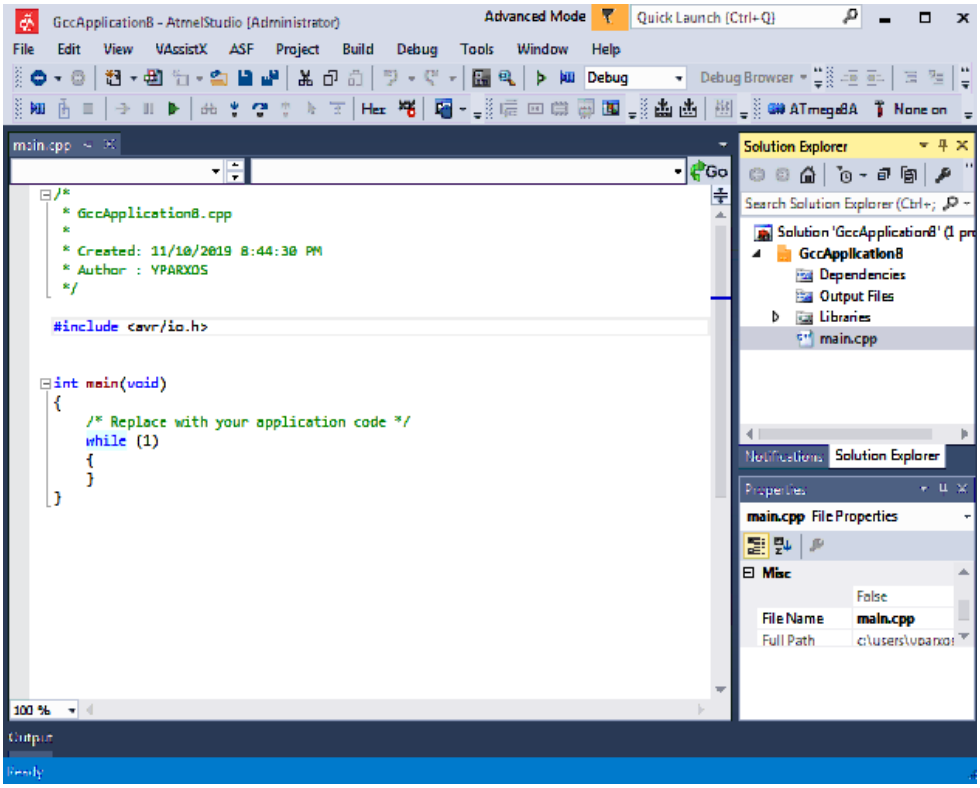
Εικόνα 2.8. *Οικογένεια Μικροελεγκτών.*



Εικόνα 2.9. *Όλοι οι μικροελεγκτές της σειράς ATmega.*

Κατόπιν και μέσα στο πλαίσιο που εμφανίζεται σύρουμε την γραμμή κύλισης βρίσκουμε τον μικροελεγκτή επιλέγουμε για τα παραδείγματα και της εργασίες που θα ακολουθήσουν τον μικροελεγκτή ATmega8A και πατάμε **OK**, αυτό ήταν! Το STUDIO 7 αμέσως μας οδηγεί σε περιβάλλον προγραμματισμού από οπού θα ξεκινήσουμε τα πρώτα βήματα εγγραφής γραμμών προγραμματισμού.

Όπως παρατηρούμε στην παρακάτω εικόνα βρισκόμαστε στο αρχικό περιβάλλον προγραμματισμού του STUDIO 7,οπου και εμφανίζεται η αρχική φόρμα εργασίας με όλα τα στοιχεία της.



Εικόνα 2.10. *Το αρχικό περιβάλλον εργασίας του ATMEL STUDIO 7*

Πριν αρχίσουμε την εγγραφή γραμμών κώδικα είναι σημαντικό να εξηγήσουμε της βασικές λειτουργίες του προγραμματιστικού περιβάλλοντος του STUDIO 7.

Ο σχολιασμός του κώδικα στον κειμενογράφο.

Στις πρώτες γραμμές που εμφανίζονται στον κειμενογράφο (editor) του περιβάλλοντος προγραμματισμού όπως παρατηρούμε μπροστά από κάθε γραμμή υπαρχή ένας αστερίσκος.

Επίσης στην αρχή της πρώτης γραμμής υπάρχει μια ανάποδη κάθετος μπροστά από έναν αστερίσκο, ενώ στο τέλος των γραμμών η ανάποδη κάθετος βρίσκεται μπροστά από τον αστερίσκο.

```
/*
 * GccApplication1.cpp
 *
 * Created: 15/4/2022 8:08:07 μμ
 * Author: GEORGE GEORGIADIS
 */
```

Αυτή η διαδικασία σηματοδότησης των γραμμών είναι υποχρεωτική εάν θέλουμε να βάλουμε σχόλια στον κώδικα μας αλλιώς ο συμβολομεταφραστής θα παράγει λάθος κατά την μετάφραση του κώδικα εάν δεν τηρηθούν τα παραπάνω.

Σημείωση: Στα σχόλια γράφουμε συνήθως το όνομα του αρχείου, την ημερομηνία δημιουργίας του,τον συγγραφέα του κώδικα,και διαφορά σχόλια για τα προβλήματα που τυχόν αντιμετωπίζουμε κατά την διάρκεια ανάπτυξης του προγράμματος μας κλπ.

1.5.4. Οδηγίες προεπεξεργαστή (Preprocessor directives)

Αμέσως παρακάτω στο παράδειγμα μας υπάρχει μια γραμμή που ονομάζεται #**include** <avr/io.h> (συμπεριλαμβάνω, περιλαμβάνω) και είναι πολύ σημαντική για τον προγραμματισμό μας καθώς μέσω αυτής της κεφαλίδας ρυθμίζονται και περιλαμβάνονται ορισμοί του συνόλου των καταχωρητών I/O και των αντίστοιχων τιμών των bit όπως καθορίζονται στην τεκμηρίωση της Microchip.

Περιλαμβάνονται επίσης τα συγκεκριμένα ονόματα που μπορούν να χρησιμοποιηθούν για ορισμούς των λειτουργιών διακοπής (Interrupt) όπως τεκμηριώνονται από την Atmel. Επίσης το αρχείο κεφαλίδας περιλαμβάνει τους κατάλληλους ορισμούς I/O για τη συσκευή που έχει καθορίσει ο διακόπτης γραμμής εντολών mCPU = μεταγλωττιστής.

Αυτό γίνεται με την εκτροπή στο κατάλληλο αρχείο **<avr / ioXXXX.h>** το οποίο δεν πρέπει ποτέ να συμπεριληφθεί άμεσα στης γραμμές κώδικα πάρα μόνο μέσω της κεφαλίδας #**include**.

Ορισμένα ονόματα αρχείων που είναι κοινά σε όλες τις συσκευές AVR ορίζονται απευθείας μέσα στο **<avr / common.h>,** το οποίο περιλαμβάνεται στο αρχείο **<avr / io.h>,** αλλά οι περισσότερες λεπτομέρειες προέρχονται από το αντίστοιχο αρχείο.

Τέλος, ορίζονται και διάφορες μακροεντολές πχ,

RAMEND, η τελευταία διεύθυνση RAM μνήμης στον μικροελεγκτή.

FLASHEND Η τελευταία διεύθυνση byte στο χώρο του προγράμματος Flash κλπ. Βέβαια θα μπορούσαν να υπάρχουν η να γράψουμε και οποιασδήποτε άλλες οδηγίες.

Παράδειγμα οδηγίας **#define.** Πρόκειται για μια μακροεντολή προεπεξεργαστή όπου μέσω αυτής μπορούμε να ορίσουμε μια σταθερά.

```
#define F_CPU 16000000UL //clock speed is 16MHz
```

Με την οδηγία αυτή ορίζουμε ότι το κεντρικό ρολόι του συστήματος ταλαντώνεται στα 16MHz.

Η αναφορά και η ανάλυση των παραπάνω γίνεται για όσους έρχονται για πρώτη φορά σε επαφή με το προγραμματιστικό περιβάλλον του AVR STUDIO 7 αλλά και για τους αρχάριους στον προγραμματισμό των μικροελεγκτών AVR.

Βέβαια σκοπός αυτού του βιβλίου δεν είναι να διδάξει συγκεκριμένη γλωσσά προγραμματισμού. Αυτό θεωρείται ότι όσοι ασχολούνται με τους μικροελεγκτές όλο και κάτι γνωρίζουν από διαφορά περιβάλλοντα και γλώσσες προγραμματισμού άλλοι περισσότερο και άλλοι λιγότερο, αλώστε υπάρχει πλούσια βιβλιογραφία για να εξασκηθεί όποιος ενδιαφέρετε για της γλώσσες προγραμματισμού των μικροελεγκτών, C++, Assembly,κ.λπ.

Συνεχίζοντας την αναφορά για την φόρμα του περιβάλλοντος προγραμματισμού και αφού τελειώσαμε με τις διάφορες οδηγίες προεπεξεργαστή (Preprocessor directives).

Η αμέσως επόμενη γραμμή που συναντάμε είναι η αρχή του κεντρικού προγράμματος, με την πρόταση **int main (void)** και από κάτω ακολουθούν οι γραμμές με το κυρίως πρόγραμμα οι οποίες ομαδοποιούνται από δεξιά και από αριστερά με τα άγκιστρα { }.

Συνοψίζοντας η οδηγία #**include** ουσιαστικά ορίζει στον συμβολομεταφραστή να συμπεριλάβει και να ενσωμάτωση τα περιεχόμενα, του καθορισμένου αρχείου στην αρχή του προγράμματος, στο παράδειγμα μας είναι το αρχείο <avr / io.h>, #include <avr / io.h>.

Όλα τα αρχεία με προέκταση ονόματος **h** τα οποία ενσωματώνουμε στην αρχή των προγραμμάτων ονομάζονται αρχεία κεφαλίδων.

Υπάρχουν διάφορα αρχεία κεφαλίδων το κάθε ένα από αυτά περιέχει ορισμούς για διαφορετικές λειτουργίες του συμβολομεταφραστή.

Αυτό που χρειάζεται ο αναγνώστης είναι να κατανόηση ότι η πρόταση #**include** μας επιτρέπει να χρησιμοποιήσουμε αυτά τα αρχεία κεφαλίδων.

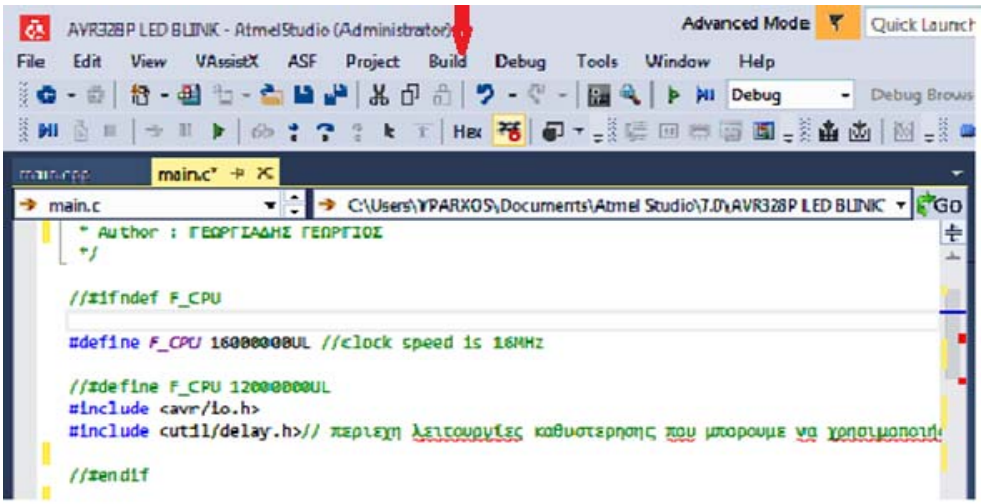
Σημείωση: Τα αρχεία με προέκταση ονόματος **h** βρίσκονται στον φάκελο εγκατάστασης του AVR STUDIO 7 και στους υποκαταλόγους **include** σε μορφή **ASCII** ώστε να έχουμε πρόσβαση για ανάγνωση η για εμφάνιση.

Είναι προφανές ότι μόλις ολοκληρώσουμε το όποιο πρόγραμμά μας θα θελήσουμε να ελέγξουμε την λειτουργία του κατά ποσό ανταποκρίνεται στο σκοπό για τον οποίο το δημιουργήσαμε εάν έχει σφάλματα κ.λπ.

Αυτό που χρειάζεται είναι να εξετάσουμε τον τρόπο με τον οποίο δημιουργούνται τα εκτελέσιμα αρχεία (Hex) από πηγαίο κώδικα στο AVR Microchip Studio 7.

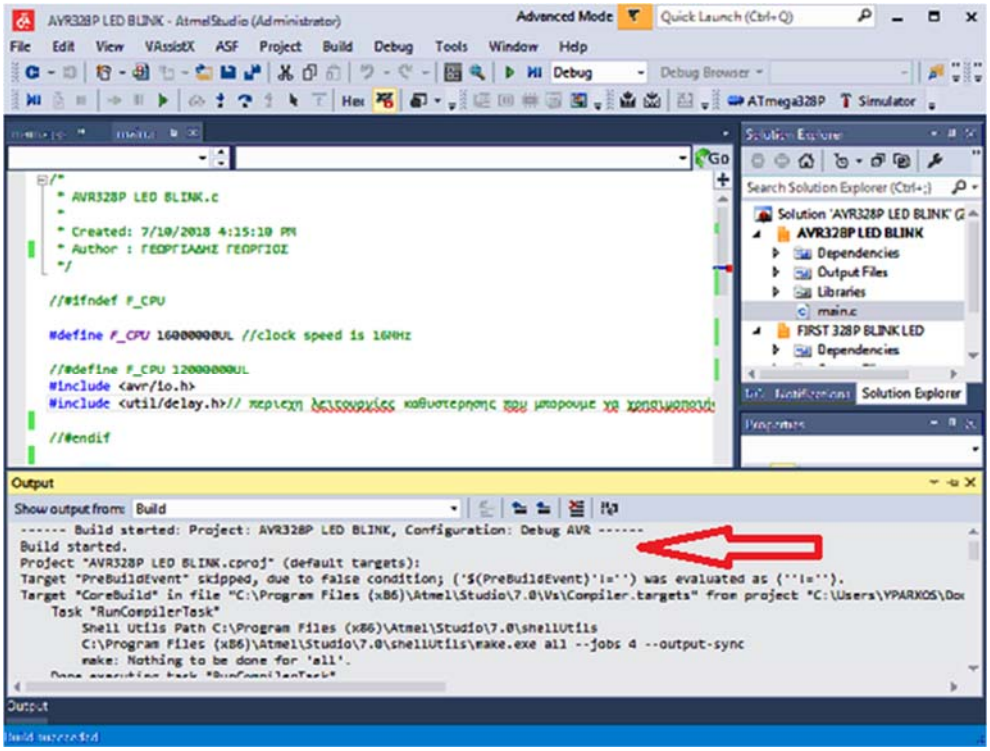
Έτσι αφού έχουμε γράψει η έχουμε εισάγει τον πηγαίο κώδικα στον κειμενογράφο του STUDIO 7, είμαστε έτοιμοι για την δημιουργία του εκτελέσιμου αρχείου της εφαρμογής με ένα απλό βήμα κάνοντας κλικ στην επιλογή **"Build"** της κεντρικής φόρμας του STUDIO7 και στο μενού που εμφανίζεται κάνουμε κλικ στην επιλογή**"Build solution" η "Rebuild solution"** εάν πρόκειται για επανάληψη.

Όπως και "BuildGccApplication1.c" εάν πρόκειται να επιλέξετε ένα συγκεκριμένο αρχείο μεταξύ πολλών άλλων αρχείων τα οποία ενδέχεται να βρίσκονται στην επάνω γραμμή της φόρμας του κειμενογράφου του STUDIO7,οπου "GccApplication1"οποιαδηποτε ονομασία έχετε στο αρχείο αυτό.



Εικόνα 2.11. *Παρατηρούμε το κόκκινο βέλος στην επιλογή "Build" που πρέπει να επιλέξουμε για να μετατρέψουμε τον πηγαίου κώδικα σε μορφή στην οποία μπορεί να εκτελεστεί το πρόγραμμα μας.*

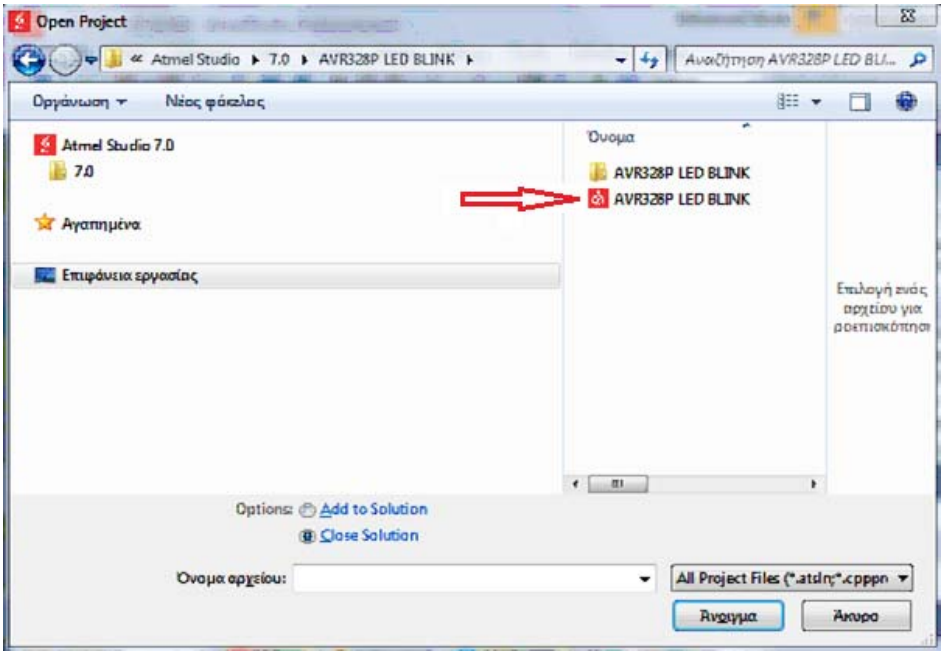
Αμέσως με τα την εντολή "Build" ξεκινά η μετατροπή του πηγαίου κώδικα όπως παρατηρούμε στην παρακάτω εικόνα στο κάτω μέρος της φόρμας προγραμματισμού.



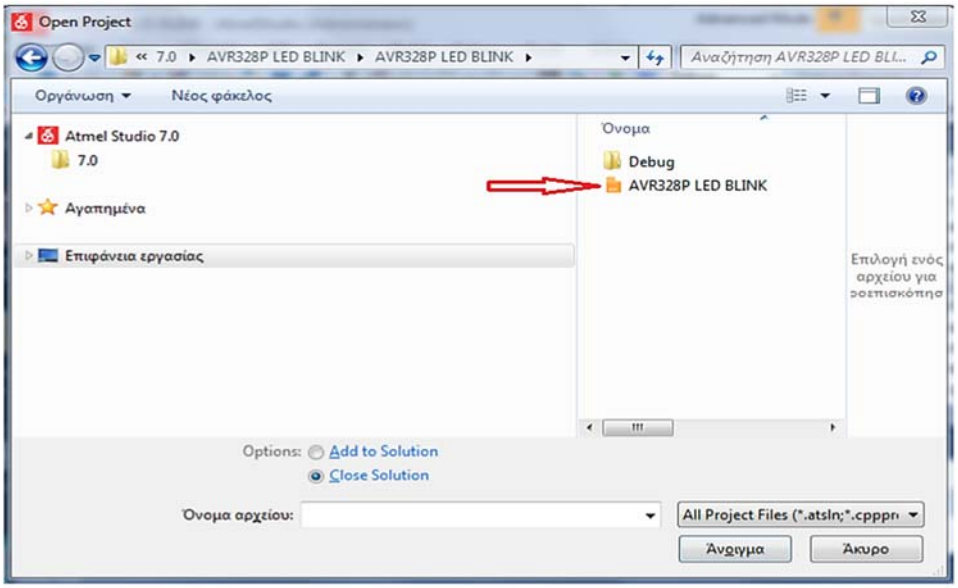
Εικόνα 2.12. *Παρατηρούμε την μετατροπή του κώδικα μας σε εκτελέσιμη μορφή.*

Στην περίπτωση που όλα πάνε κατ'ευχήν, τότε και μέσα στον κατάλογο που δημιουργήσαμε δημιουργείται ένας άλλος κατάλογος μαζί με το αρχείο του πηγαίου κώδικα **(solution file)** με την ίδια ονομασία του καταλόγου που δημιουργήσαμε στην αρχή κατά την διάρκεια δημιουργίας του νέου αρχείου μας. Μέσα σε αυτόν τον νέο κατάλογο υπαρχή το αρχείο Project file και ένας υποκατάλογος με την ονομασία **(Debug)** όπου μέσα σε αυτόν υπάρχουν τα παραχθέντα αρχεία από τον πηγαίο κώδικα συμπεριλαμβανομένου και του εκτελέσιμου αρχείου Hex.

Εάν υπάρχουν λάθη στον προγραμματισμό η στο συντακτικό του κώδικα αυτά θα εμφανισθούν στην έξοδο (Output) του συμβολομεταφραστή περιοχή που υποδεικνύει το κόκκινο βέλος, με αποτέλεσμα την ελλειπή η την μη μετατροπή του κώδικα σε άλλη μορφή αρχείων.



Εικόνα 2.13. *Ο κατάλογος AVR328P LED BLINK μαζί με το αρχείο του πηγαίου κώδικα (solution file).*



Εικόνα 2.14. *Η εμφάνιση του υποκαταλόγου AVR328P LED BLINK με το αρχείο Project που εμπεριέχει εκτός των άλλων και το εκτελέσιμο αρχείο Hex.*

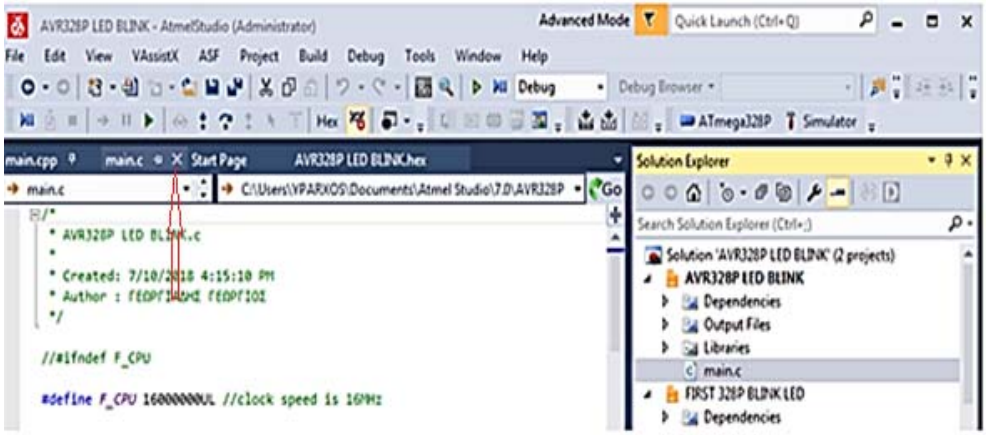
Επίσης εάν θελήσουμε να ανοίξουμε ένα είδη υπάρχον αρχείο στον κειμενογράφο προγραμματισμού του STUDIO7 ακολουθούμε την διαδικασία

File → Open → File.

Η εάν θέλουμε να επεξεργαστούμε ένα Project file επιλέγουμε

File → Open → Project/Solution.

Για το κλείσιμο ενός αρχείου επιλέγουμε **File-Close**,η με τον δείκτη του ποντικιού επιλέγουμε **X** στο όνομα του αρχείου που εμφανίζεται στην επάνω ραβδογραμμή του κειμενογράφου προγραμματισμού του STUDIO7 Εικόνα 2.15.

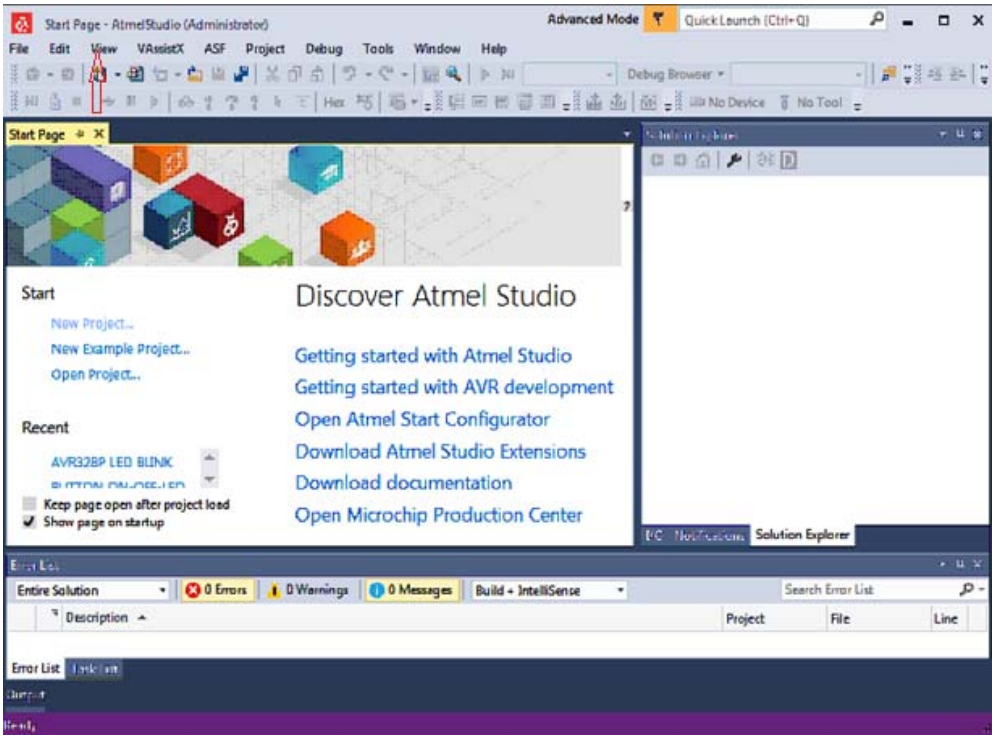


Εικόνα 2.15. *Για το κλείσιμο ενός αρχείου επιλέγουμε* ***X*** *στο όνομα του αρχείου όπως υποδεικνύει το κόκκινο βέλος.*

Σημείωση: στην περίπτωση κατά την οποία δεν δημιουργήσουμε κανένα αρχείο όταν εκτελέσουμε το STUDIO 7 η κεντρική οθόνη εμφανίζει την σελίδα της εφαρμογής (Start Page) αρχική σελίδα.

Αυτό βέβαια αν θελήσουμε αργότερα το απενεργοποιούμε αποεπιλέγοντας το τετραγωνάκι που βρίσκεται στην περιοχή κάτω αριστερά της αρχικής σελίδας (Start Page).

Επίσης έχουμε την δυνατότητα ανά πασά στιγμή να ανοίγουμε και να κλείνουμε την αρχική σελίδα (Start Page) μέσα από την επιλογή **View,** και στο μενού που θα εμφανισθεί επιλέγουμε **Start Page.**



Εικόνα 2.16. *Η αρχική σελίδα* ***(Start Page)*** *του STUDIO 7*

Κάπου εδώ τελειώσαμε με την εγκατάσταση της εφαρμογής, την παρουσίαση δημιουργίας και αποθήκευσης αρχείων όπως και την περιληπτική λειτουργία του προγραμματιστικού περιβάλλοντος του STUDIO 7.

Μην ανησυχείτε καθώς κατά την διάρκεια δημιουργίας και εξέτασης των εφαρμογών μας, τα εργαλεία καθώς και οι διάφορες λειτουργίες του STUDIO7 που θα συναντούμε θα εξετάζονται αναλόγως.

1.6. Υλικό μέρος (Hardware)

Ως συστατικό μέρος μιας λειτουργικής μονάδας, είναι απαραίτητο και το υλικό μέρος αυτής, ώστε να έχουμε την δυνατότητα ελέγχου και εκτέλεσης του λογισμικού που δημιουργούμε.

Σήμερα υπάρχουν για αυτόν τον σκοπό δεκάδες αναπτυξιακές κάρτες με της οποίες μπορούμε γρηγορά και αποτελεσματικά να ελέγξουμε τα προγράμματα μας.

Τα βασικά συστατικά που μπορεί να διαθέτουν είναι μια η περισσότερες μονάδες επικοινωνίας π.χ σειριακή μονάδα RS232 η μια θύρα USB, και SPI για το φόρτωμα

των εφαρμογών στον μικροελεγκτή, διάφορους αισθητήρες, ακροδέκτες εξόδου για εξωτερικές διασύνδεσης, ενσωματωμένα LED και διακόπτες (buttons) επάνω στην αναπτυξιακή κάρτα κλπ.

Η ATMEL και τώρα η MICROCHIP έχουν ανάπτυξη και προσφέρουν διάφορες αναπτυξιακές μονάδες καρτών.

Θα αναφέρουμε μερικά μοντέλα της Atmel όπως της κάρτες **STK500-600, DRAGON.** Και Microchip όπως η σειρά **Xplained Pro Evaluation Board**, οι οποίες κατασκευάζονται από την ιδιά την εταιρία που κατασκευάζει και τους μικροελεγκτές. Φυσικά στην αγορά υπάρχει πληθώρα αναπτυξιακών καρτών και εργαλείων από τρίτους κατασκευαστές για όλους τους μικροελεγκτές AVR.

Η επιλογή μιας αναπτυξιακής κάρτας εξαρτάται από διάφορους παράγοντες όπως:

- Η απόδοση της αναπτυξιακής κάρτας
- τα ενσωματωμένα χαρακτηριστικά που διαθέτη, διακόπτες, LED, LCD DISPLAY, ακροδέκτες (PIN) εισόδου - εξόδου και φυσικά,
- ανάλογα με την τιμή της.

Επίσης θα χρειαστούμε και έναν προγραμματιστή (Programmer) για να φορτώνουμε της εφαρμογές μας (αρχεία Hex) στον μικροελεγκτή που θα της εκτελεί.

1.7. Βασικά εργαλεία για την κατασκευή και ανάπτυξη των εφαρμογών μας

Για την υλοποίηση των κατασκευών φυσικά και θα χρειαστούμε διάφορα εργαλεία εργαστήριου όπως:

- Όργανα μέτρησης τάσης, έντασης, κ.λ.π.
- Όργανα τροφοδοσίας, όργανα παραγωγής παλμών (Γεννήτρια).κ.λ.π.
- Ένα ψηφιακό παλμογράφο.
- Ένα σύνολο από βιδολόγους διαφόρων μεγεθών (κατσαβίδια).
- Ένα κόφτη ιδιαίτερα λεπτό μιας και χρησιμοποιούμε για αυτού του είδους της εργασίες λεπτούς ακροδέκτες και καλώδια για εύκολη κοπή και απογύμνωση.
- Μια γενική πένσα για διάφορες συσφίξεις, και ένα μυτοτσίμπιδο για το λύγισμα των ακροδεκτών των καλωδίων, και για τον χειρισμό των διαφόρων εξαρτημάτων της κατασκευής μας.

- Επίσης καλό θα ήταν να έχουμε και ένα απογυμνωτή καλωδίων με τον όποιον μπορούμε να αφαιρέσουμε το τμήμα της πλαστικής μόνωσης των καλωδίων.
- Ένα εργαλείο συγκολλήσεων το γνωστό μας κολλητήρι περίπου στα 30W - 40W και με μύτες διαφόρων τύπων και μεγεθών ως προς την λεπτότητα της μύτης.
- Βέβαια ιδανική λύση θα ήταν να έχουμε στην κατοχή μας ένα σταθμό συγκόλλησης και αποκόλλησης εξαρτημάτων.
- Εναλλακτικά θα πρέπει να έχουμε ένα εργαλείο ελεγχόμενης θερμοκρασίας αέρα περίπου στους 0-εως 450K βαθμούς κελσίου.
- Ένα πιστόλι η σταθμό αναρρόφησης του υλικού κόλλησης (καλαϊ) για την αποκόλληση των εξαρτημάτων.
- Εναλλακτικά μια απλή τρόμπα αναρρόφησης.
- Ένα μικρό χειροκίνητο ηλεκτρικό δράπανο μαζί με ένα σύνολο από διαφόρων μεγεθών τρυπάνια για την διάνοιξη των οπών στις πλακέτες και στα διάφορα κουτιά των κατασκευών μας.

Εικόνα 2.17. *Set από διαφορά εργαλεία χειρός.*

***Εικόνα 2.18.** κλασικό κολλητήρι χειρός.*

***Εικόνα 2.19.** Ψηφιακό πολύμετρο*

Εικόνα 2.20. *Ψηφιακός Παλμογράφος.*

- Προαιρετικά ένα Ψηφιακό Παλμογράφο εάν διαθέτετε τα κατάλληλα χρήματα.
- Μια μικρή μέγγενη για την συγκράτηση των πλακετών και των διαφόρων εξαρτημάτων κατά την επεξεργασία των,όπως άνοιγμα οπών,τοποθέτηση, κλπ.
- Επίσης για την κατασκευή των διαφόρων πρωτότυπων κατασκευών μας θα χρειαστούμε μια σειρά από διάφορα μεγέθη πλακέτες χαλκού (Board PCB).
- Διάτρητες πλακέτες βακελίτη για τους πειραματισμούς μας σε διάφορες εκδόσεις.
- Η πλακέτες αυτές είναι εφοδιασμένες με πολλές οπές οι οποίες έχουν ίση απόσταση μεταξύ τους και είναι κατασκευασμένες με μονή όψη δηλαδή διαθέτουν τυπωμένες χάλκινες επαφές μόνο από την μια πλευρά τους.

Εικόνα 2.21. *Ψηφιακό βολτόμετρο.*

***Εικόνα 2.22.** Φωτογραφία από διάτρητες πλακέτες διαφόρων μεγεθών.*

Ένα πρωτότυπο ράστερ για την αρχική πειραματική εφαρμογή των κυκλωμάτων μας, για όσους δεν γνωρίζουν το ράστερ είναι μια ιδιαίτερη κατασκευή από πλαστικό υλικό η όποια έχει ειδικές οπές κατάλληλης διαμέτρου με ίσες απόστασης μεταξύ των και με σχετική διάταξη σε μορφή πίνακα κελίων.

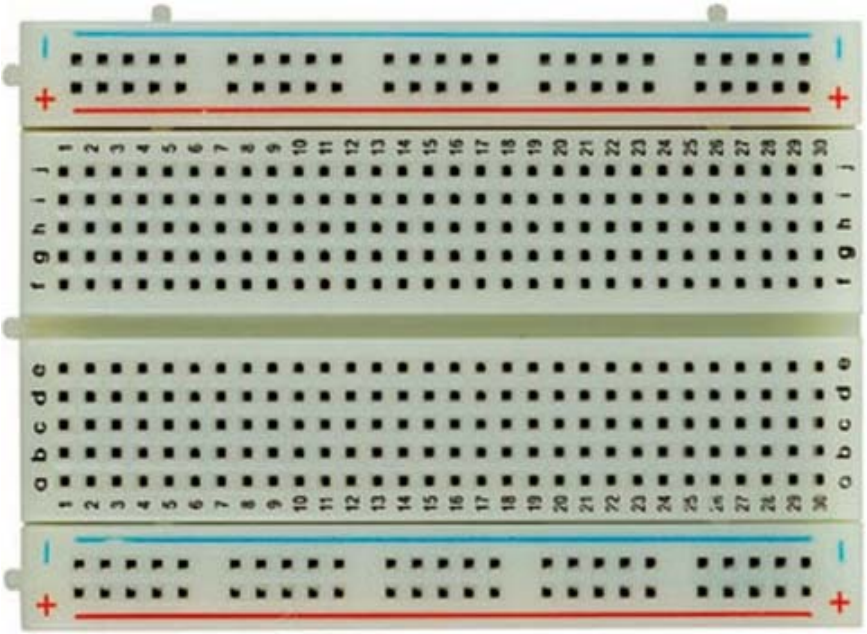
Οι οπές που ανήκουν στην ιδιά γραμμή της παραπάνω διάταξης είναι εσωτερικά συνδεδεμένες μεταξύ τους.

Επίσης στην δεξιά και αριστερή πλευρά ενός ράστερ υπάρχουν δυο μεγάλες σειρές από οπές οι οποίες και είναι συνδεδεμένες μεταξύ τους εσωτερικά ξεχωριστά για την δεξιά πλευρά και ξεχωριστά για την αριστερή πλευρά οι οποίες χρησιμοποιούνται ως κοινές γραμμές τροφοδοσίας του κυκλώματος + **συν** για την μια πλευρά και – **πλην** για την άλλη πλευρά.

Στην αγορά υπάρχουν ράστερ που έχουν + **συν** και – **πλην** και στις δυο πλευρές για την διευκόλυνση του χρήστη. Όπως φαίνεται στην παρακάτω εικόνα.

Σημείωση: Η διάμετρος των οπών είναι συγκεκριμένη ώστε να εφαρμόζουν καταλληλά οι ακροδέκτες των διαφόρων ηλεκτρονικών εξαρτημάτων όπως είναι τα τρανζίστορ, οι δίοδοι, οι αντίστασης, οι πυκνωτές κλπ.

Συμπερασματικά θα μπορούσαμε να πούμε ότι το ράστερ αποτελεί ένα απλό εύκολο και ανέξοδο τρόπο με τον οποίο μπορούμε να ελέγξουμε την λειτουργία των πειραματικών εφαρμογών των κυκλωμάτων μας.



Εικόνα 2.23. *Φωτογραφία τυπικής διάταξης ράστερ με διατάξει θετικής* **+** ***συν*** *και αρνητικής* **–** ***πλην*** *τροφοδοσίας στης αμφότερες πλευρές του.*