ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΜΕΛΕΤΗ ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΑΥΤΟΜΑΤΙΣΜΟΥ ΕΛΕΓΧΟΜΕΝΟΥ ΑΠΟ ΤΟ ΔΙΑΔΥΚΤΙΟ ΓΙΑ ΑΠΟΜΑΚΡΥΣΜΕΝΗ ΕΠΙΒΛΕΨΗ ΧΩΡΟΥ

Επιβλέπων: Δρ. Χατζάκης Ιωάννης
Επίκουρος Καθηγητής Τμήματος Ηλεκτρονικών Μηχανικών Τ.Ε.
Αποστολόπουλος Δημήτρης Α.Μ. : 4169
Άννα-Μαρία Γιαννακοπούλου Α.Μ. : 3952

Χανιά, 2014

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θα θέλαμε να αναγνωρίσουμε τις απαραίτητες γνώσεις που αποκτήσαμε κατά την φοίτηση μας στο Τμήμα Ηλεκτρονικών Μηχανικών του ΤΕΙ Κρήτης όσον αφορά αναλογικά καθώς και ψηφιακά συστήματα, στις διαδικασίες μελέτης και σχεδιασμού συστημάτων καθώς και στην διαδικασία προγραμματισμού αυτών. Θα θέλαμε να ευχαριστήσουμε επίσης τον Κύριο Ρηγάκη Ηρακλή (Επιστημονικό Συνεργάτη του Τμήματος Ηλεκτρονικών) καθώς και τον υπεύθυνο της πτυχιακής μας Κύριο Χατζάκη Ιωάννη (Επίκουρο Καθηγητή του Τμήματος Ηλεκτρονικών Μηχανικών) για την πολύτιμη βοήθεια που μας προσέφεραν στην σχεδίαση, υλοποίηση και προγραμματισμό των συστημάτων που κατασκευάσαμε παρέχοντας μας την απαραίτητη βοήθεια σε τεχνικές δυσκολίες που αντιμετωπίσαμε τόσο κατασκευαστικά όσο και σε επίπεδο προγραμματισμού.

ΠΕΡΙΛΗΨΗ ΠΤΥΧΙΑΚΗΣ

Στην πτυχιακή αυτή σκοπός μας ήταν η σχεδίαση, προγραμματισμός και υλοποίηση ενός συστήματος που θα είχε σαν στόχο την επίβλεψη ενός χώρου με την βοήθεια αισθητήρων τον έλεγχο τυχόν παραβίασης αυτού και ενημέρωση μίας διεπαφής χρήστη προσβάσιμη από το διαδίκτυο η οποία σχεδιάστηκε εξολοκλήρου από την αρχή για τις ανάγκες του συγκεκριμένου έργου.

ABSTRACT

In this thesis our aim was to design, planning and implementation of a system that would like to oversee a space using sensors to check for violations thereof, and an updated user interface accessible from the internet which was designed from scratch for the needs of the project.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΕΡΓΑΣΙΑΣ

Η εργασία αυτή υλοποιήθηκε σε διάφορα στάδια τα οποία περιελάμβαναν σχεδιασμό, προγραμματισμό και κατασκευή. Το κάθε στάδιο αποτελείται από επιμέρους κομμάτια που σχετικά με την κεντρική σχεδίαση, τον προγραμματιστή, το στοιχείο του XPORT-05R που περιλαμβάνει την θύρα RJ45 για την σύνδεση διαδικτύου με την κατασκευή μας αλλά και σε όλα τα επιμέρους στοιχεία τα οποία έπρεπε να σχεδιαστούν και να κατασκευαστούν προκειμένου να υλοποιηθεί η όλη κατασκευή.

1.2 Η ΕΝΝΟΙΑ ΤΟΥ ΑΥΤΟΜΑΤΙΣΜΟΥ

Ο αυτοματισμός [1] ερευνά τη συμπεριφορά δυναμικών συστημάτων μοντελοποιώντας τα με τα μεθοδολογικά και μαθηματικά εργαλεία της επεξεργασίας σήματος. Έτσι μεταχειρίζεται τα συστήματα ως μαύρα κουτιά με είσοδο και έξοδο. Ως είσοδος θεωρείται ένα σήμα, αναλογικό ή ψηφιακό, συλλεγόμενο από κάποιο σημείο του συστήματος. Τα ενδιάμεσα κουτιά αναπαριστούν τις διάφορες διαταράξεις που επηρεάζουν το σήμα, όπως τριβές στους ενεργοποιητές, αποτέλεσμα των στοιχείων του ελέγχου που παρεμβάλλονται, τους ελεγκτές.
Αυτά τα αποτελέσματα συνήθως αναπαρίστανται με μαθηματικές συναρτήσεις, τις συναρτήσεις μεταφοράς. Μία συνάρτηση μεταφοράς προσδιορίζει ένα σύστημα και τον τρόπο που μεταβάλλει κάθε σήμα εισόδου. Η έξοδος του συστήματος ονομάζεται ***αναφορά*** και ανταποκρίνεται στην τιμή του σήματος κατόπιν ενεργοποίησης των προηγούμενων συναρτήσεων μεταφορών σε αυτήν. Όταν μια ή περισσότερες μεταβλητές εξόδου ενός συστήματος πρέπει να ακολουθήσουν την τιμή κάποιας αναφοράς που μεταβάλλεται με τον χρόνο, χρειάζεται να προστεθεί

ένας ελεγκτής που να χειρίζεται τις τιμές των σημάτων εισόδου έως ότου επιτευχθεί η επιθυμητή έξοδος.

1.3 Η ΕΝΝΟΙΑ ΤΟΥ ΔΙΑΔΥΚΤΙΟΥ

Το **Διαδίκτυο** (Internet) [2] είναι παγκόσμιο σύστημα διασυνδεδεμένων δικτύων υπολογιστών, οι οποίοι χρησιμοποιούν καθιερωμένη ομάδα πρωτοκόλλων, η οποία συχνά αποκαλείται "TCP/IP" (αν και αυτή δεν χρησιμοποιείται από όλες τις υπηρεσίες του Διαδικτύου) για να εξυπηρετεί εκατομμύρια χρηστών καθημερινά σε ολόκληρο τον κόσμο. Οι διασυνδεδεμένοι ηλεκτρονικοί υπολογιστές ανά τον κόσμο, οι οποίοι βρίσκονται σε ένα κοινό δίκτυο επικοινωνίας, ανταλλάσσουν μηνύματα (πακέτα) με τη χρήση διαφόρων πρωτοκόλλων (τυποποιημένοι κανόνες επικοινωνίας), τα οποία υλοποιούνται σε επίπεδο υλικού και λογισμικού. Το κοινό αυτό δίκτυο καλείται Διαδίκτυο.

1.4 Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA

Η επίσημη εμφάνιση της *Java* [3] αλλά και του *HotJava* (πλοηγός με υποστήριξη *Java*) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η *Sun* την ανακοίνωσε στο συνέδριο *Sun World* 1995. Ο πρώτος μεταγλωττιστής(*compiler*) της ήταν γραμμένος στη γλώσσα C από τον James Gosling. Το 1994, ο A.Van Hoff ξαναγράφει τον μεταγλωττιστή της γλώσσας σε *Java*, ενώ το Δεκέμβριο του 1995 πρώτες οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια να χρησιμοποιήσουν τη *Java* για την δημιουργία λογισμικού. Από εκεί και πέρα η Java ακολουθεί μία ανοδική πορεία και είναι πλέον μία από τις πιο δημοφιλείς γλώσσες στον χώρο της πληροφορικής.

❖ Τα χαρακτηριστικά της JAVA

Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε *Java* τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (καθώς και σε κονσόλες παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και λειτουργικού συστήματος (Windows, Unix, Linux, BSD, MacOS). Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Ο συμβολικός κώδικας (*assembly*) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της *Εικονικής Μηχανής* (*Virtual Machine* ή VM).

❖ Η Εικονική Μηχανή της JAVA

Αφού γραφεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία «.class». Στη συνέχεια τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί (να σημειωθεί εδώ ότι αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή (Virtual Machine). Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με

αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java.
Πρέπει να σημειωθεί ότι η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ. Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

- **Επιδόσεις**

Παρόλο που η εικονική μηχανή προσφέρει όλα αυτά (και όχι μόνο) τα πλεονεκτήματα, η *Java* αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου (high-level) όπως η C και η C++. Εμπειρικές μετρήσεις στο παρελθόν είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από την Java. Ωστόσο γίνονται προσπάθειες από τη Sun για τη βελτιστοποίηση της εικονικής μηχανής, ενώ υπάρχουν και άλλες υλοποιήσεις της εικονικής μηχανής από διάφορες εταιρίες (όπως της IBM), οι οποίες μπορεί σε κάποια σημεία να προσφέρουν καλύτερα και σε κάποια άλλα χειρότερα αποτελέσματα. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ. Οι τελευταίες εκδόσεις του javac με τη χρήση της τεχνολογίας Hot Spot έχουν καταφέρει αξιόλογες επιδόσεις που πλησιάζουν ή και ξεπερνούν σε μερικές περιπτώσεις τον εγγενή κώδικα.

1.5 Ο ΔΙΑΥΛΟΣ I²C

Ο δίαυλος **I²C** [4] είναι ένας σειριακός δίαυλος που δημιουργήθηκε από τη Philips και χρησιμοποιείται για την σύνδεση περιφερειακών μικρής ταχύτητας σε motherboard, embedded systems, κινητά τηλέφωνα ή άλλες ηλεκτρονικές συσκευές. Ο δίαυλος I2C δεν χρησιμοποιείται μόνο για την επικοινωνία συσκευών που βρίσκονται πάνω σε ένα τυπωμένο κύκλωμα, αλλά και για την επικοινωνία συσκευών που συνδέονται με καλώδια.

Χρησιμοποιεί μόνο δύο καλώδια, τα οποία είναι αμφίδρομης κατεύθυνσης: Τα SCL και SDA. Η γραμμή SCL είναι η γραμμή ρολογιού, ενώ η SDA είναι η γραμμή δεδομένων. Οι γραμμές αυτές συνδέονται σε όλες τις συσκευές στον δίαυλο I²C. Προφανώς εκτός από τα παραπάνω καλώδια, απαιτείται και ένα τρίτο καλώδιο, το οποίο είναι η γείωση (GND) ή 0 V. Επίσης μπορεί να υπάρχει και ένα τέταρτο καλώδιο το οποίο είναι η γραμμή τροφοδοσίας, με την οποία τροφοδοτούνται με ισχύ οι διάφορες συσκευές που υπάρχουν στο δίκτυο. Τυπικές τάσεις που χρησιμοποιούνται στο δίαυλο είναι τα +5V ή 3,3V, αν και επιτρέπονται συστήματα με διαφορετικές τάσεις (συνήθως στην περιοχή από 1,2V-5,5V).

Ο μέγιστος αριθμός κόμβων, που μπορούν να συνδεθούν στον δίαυλο, περιορίζεται από τον αριθμό των διαθέσιμων διευθύνσεων (θα επεξηγηθεί παρακάτω), αλλά και από τη συνολική χωρητικότητα του διαύλου, η οποία π.χ. για τον standard mode δεν πρέπει να υπερβαίνει τα 400pF, το οποίο και περιορίζει τις πρακτικές αποστάσεις επικοινωνίας.

Αμφότερες οι γραμμές SCL και SDA είναι τύπου *ανοικτού απαγωγού* (open drain) ή *ανοικτού συλλέκτη* (open collector στον κόσμο των TTL). Αυτό σημαίνει ότι και οι δύο αυτές γραμμές πρέπει να συνδέονται η κάθε μία, μόνο οι γραμμές και όχι κάθε συσκευή που συνδέεται στο δίαυλο ξεχωριστά, με μία αντίσταση στην γραμμή τροφοδοσίας, όπως φαίνεται στο παρακάτω σχήμα. Η αντίσταση αυτή ονομάζεται *αντίσταση τερματισμού*.

Η τιμή των αντιστάσεων δεν είναι κρίσιμη, αλλά μαζί με την χωρητικότητα του διαύλου, επηρεάζει την μέγιστη ταχύτητα λειτουργίας του διαύλου. Μεγάλες χωρητικότητες του διαύλου, μπορούν να αντισταθμιστούν με μικρές αντιστάσεις

τερματισμού. Συνηθισμένες τιμές αντιστάσεων είναι από 1ΚΩ έως 10ΚΩ. Οι αντιστάσεις αυτές δεν μπορούν να απουσιάζουν, διότι τότε οι γραμμές θα είναι μονίμως σε κατάσταση λογικού 0 και δίαυλος δεν θα δουλεύει.

- **Λογικές στάθμες των γραμμών SCL και SDA**

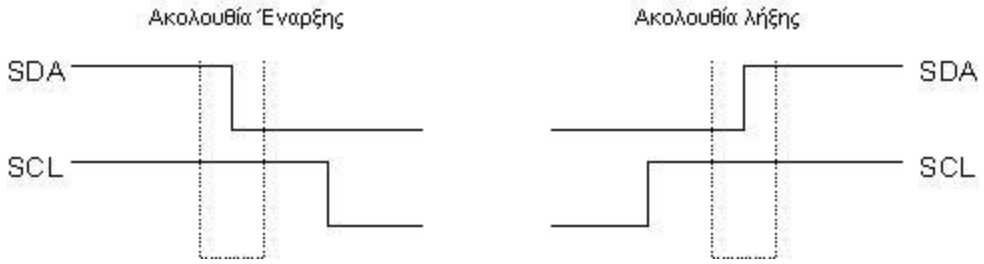
Επειδή στον δίαυλο I2C συνδέεται ποικιλία συσκευών διαφόρων τεχνολογιών (CMOS, NMOS, Διπολικής τεχνολογίας), οι οποίες μπορεί να έχουν διαφορετικές τάσεις λειτουργίας, οι στάθμες του λογικού 0 (Low) και λογικού 1 (High) σε όλες τις νέες συσκευές δεν είναι σταθερές, αλλά εξαρτώνται από την τάση τροφοδοσίας. Έτσι τα κατώφλια του *λογικού 0* και του *λογικού 1* τοποθετούνται στο 30% και το 70% της τάσης τροφοδοσίας αντίστοιχα. Δηλ. μία τάση στην περιοχή 0-0,3V_{DD}θεωρείται λογικό 0 (Low), ενώ μία τάση στην περιοχή 0,7V_{DD} έως V_{DD}, θεωρείται λογικό 1 (High). Παλαιότερα τα κατώφλια του λογικού 0 και 1 είχαν τοποθετηθεί στα 1,5V και 3,0V αντίστοιχα.

- **Κύριοι και υποτελείς (Masters and Slaves)**

Οι συσκευές στον δίαυλο I²C είναι είτε *Κύριοι (Masters)* είτε *Υποτελείς (Slave)*. Η Master συσκευή είναι αυτή που ελέγχει και οδηγεί τη γραμμή ρολογιού SCL (παράγει τους παλμού ρολογιού). Οι Slave συσκευές είναι αυτές που ανταποκρίνονται στις συσκευές Master. Μία συσκευή Slave δεν μπορεί να ξεκινήσει μία μεταφορά πάνω στο δίαυλο, μόνο μία συσκευή Master μπορεί. Σε έναν δίαυλο μπορεί να είναι συνδεμένες πολλές Master και πολλές Slaves συσκευές. Και οι Master και οι Slave συσκευές μπορούν να μεταφέρουν δεδομένα στον δίαυλο, αλλά μόνο οι Master συσκευές ελέγχουν την μεταφορά.

❖ Το φυσικό πρωτόκολλο του διαύλου I²C

Όταν μία Master συσκευή επιθυμεί να επικοινωνήσει με μία Slave συσκευή, ξεκινά στέλνοντας στον δίαυλο μία **ακολουθία έναρξης (start sequence)**. Η ακολουθία έναρξης, είναι μία από τις δύο ειδικές ακολουθίες που ορίζονται στο δίαυλο I²C, ή άλλη είναι η **ακολουθία λήξης (stop sequence).**



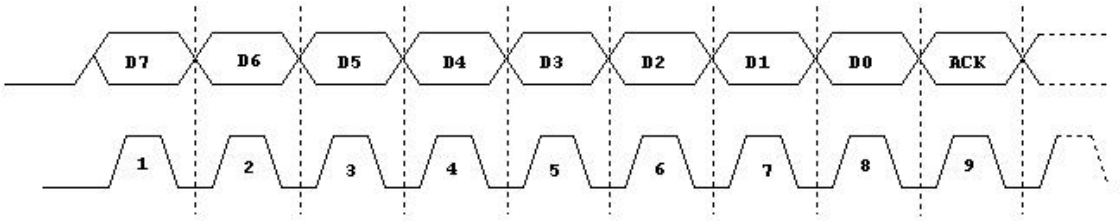
Σχήμα 1..5. Error! No text of specified style in document.**..1: Ακολουθία Έναρξης και Λήξης**

Οι ακολουθίες έναρξης και λήξης διαφέρουν στο ότι είναι οι μοναδικές θέσεις στις οποίες επιτρέπεται να αλλάζει η γραμμή δεδομένων (SDA), ενόσω η γραμμή ρολογιού είναι σε κατάσταση λογικού 1 (high).

Όταν μεταφέρονται δεδομένα, η γραμμή SDA πρέπει να παραμένει σταθερή και να μην αλλάζει όσο η γραμμή ρολογιού είναι high. Οι ακολουθίες έναρξης και λήξης σημαδεύουν την έναρξη και τη λήξη μιας μεταφοράς με μία slave συσκευή. Δηλαδή ο δίαυλος θεωρείται ότι είναι απασχολημένος, μετά από μία ακολουθία έναρξης και ελεύθερος λίγο χρόνο μετά την ακολουθία λήξης.

Τα δεδομένα μεταφέρονται σε ακολουθίες των 8 bit. Τα bit τοποθετούνται στη γραμμή SDA, ξεκινώντας από το περισσότερο σημαντικό bit (MSB). Η γραμμή SCL μετά πάλλεται high μετά low. Για κάθε 8 bit δεδομένων που μεταφέρονται, η συσκευή που λαμβάνει στέλνει πίσω ένα bit επιβεβαίωσης (ACK). Έτσι στην πραγματικότητα απαιτούνται 9 παλμοί ρολογιού, για την μεταφορά των 8 bit κάθε byte δεδομένων. Εάν η συσκευή που λαμβάνει, στείλει πίσω ένα low *bit επιβεβαίωσης (ACK)*, τότε έχει λάβει τα δεδομένα και είναι έτοιμη να λάβει το

επόμενο byte δεδομένων. Εάν στείλει πίσω ένα high bit επιβεβαίωσης (που συμβολίζεται και με NACK, από τη φράση Not Acknowledged), αυτό δείχνει ότι η συσκευή που λαμβάνει, δεν μπορεί να λάβει περαιτέρω δεδομένα και η master συσκευή πρέπει να τερματίσει την αποστολή δεδομένων, εκπέμποντας μία ακολουθία λήξης.



Σχήμα 1.5. Error! No text of specified style in document.**.2 Ακολουθία Διεύθυνσης**

Όταν στέλνουμε την διεύθυνση **A6A5A4A3A2A1A0** των 7–bit της συσκευής με την οποία θέλουμε να επικοινωνήσουμε, ακόμη και τότε στέλνουμε 8 bit. Το επιπλέον (R/W) bit χρησιμεύει να πληροφορήσει την slave συσκευή, εάν η master συσκευή πρόκειται να γράψει ή να διαβάσει από αυτήν. Εάν το bit είναι μηδέν η master συσκευή πρόκειται να γράψει. Εάν είναι 1 (ένα) πρόκειται να διαβάσει από την slave. Τα 7 bit της διεύθυνσης τοποθετούνται στα 7 πάνω bit του byte, ενώ bit ανάγνωσης/εγγραφής (R/W) στο λιγότερο σημαντικό bit (LSB), όπως απεικονίζεται στο παρακάτω σχήμα.

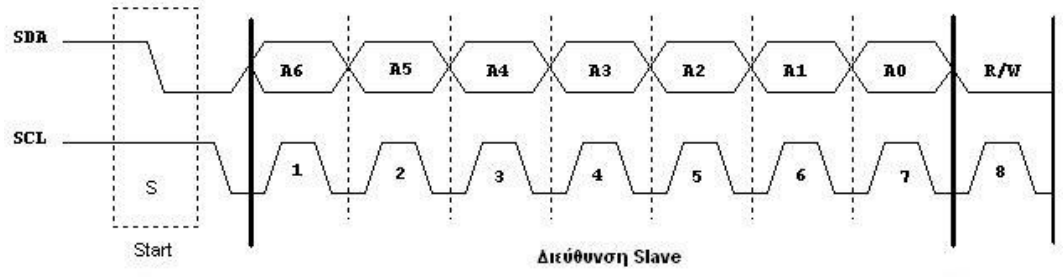
- **Εγγραφή σε μία slave συσκευή:**

Για να ξεκινήσει μία επικοινωνία, το πρώτο πράγμα που θα κάνει η master συσκευή είναι να εκπέμψει την ακολουθία έναρξης. Αυτό ειδοποιεί όλες τις slave συσκευές στο δίαυλο, ότι πρόκειται να ξεκινήσει μία εκπομπή και να ακούσουν για την περίπτωση που είναι γι' αυτές. Μετά η master συσκευή θα εκπέμψει την διεύθυνση της συσκευής. Η slave συσκευή που η διεύθυνσή της ταιριάζει με αυτήν θα συνεχίσει, ενώ όλες οι άλλες θα περάσουν σε αναμονή περιμένοντας την επόμενη επικοινωνία.

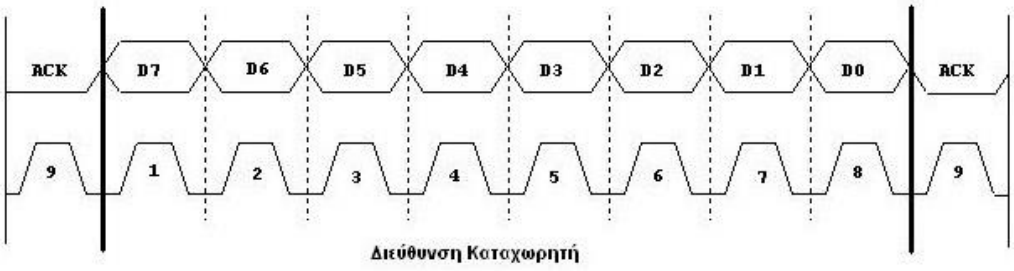
Μετά την αποστολή της διεύθυνσης της slave συσκευής, η master στέλνει τη διεύθυνση του καταχωρητή της slave στον οποίο θέλει να γράψει. Τώρα η master μπορεί να στείλει το Byte ή τα Bytes δεδομένων. Η master μπορεί να συνεχίσει να στέλνει δεδομένα, τα οποία θα τοποθετηθούν στις επόμενες θέσεις, επειδή η slave συσκευή θα αυξάνει αυτόματα την διεύθυνση του εσωτερικού καταχωρητή μετά την λήψη κάθε byte. Όταν η master συσκευή στείλει όλα τα δεδομένα, σταματάει την εκπομπή εκπέμποντας μια ακολουθία λήξης. Συγκεκριμένα τα βήματα που ακολουθούνται έχουν ως εξής:

Η master συσκευή:

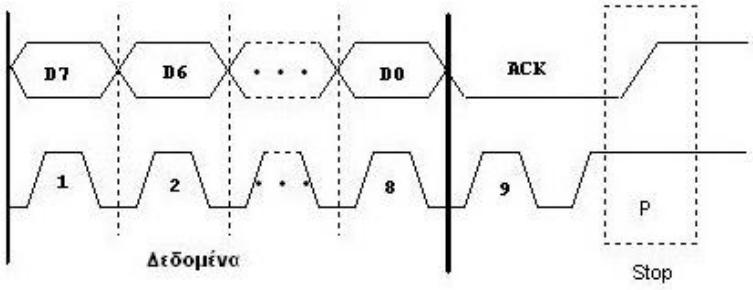
- Στέλνει την ακολουθία έναρξης
- Στέλνει την διεύθυνση της slave συσκευής με το **R/W bit low** (άρτια διεύθυνση), δηλώνοντας έτσι ότι θέλει να κάνει εγγραφή δηλ. να στείλει δεδομένα.
- Στέλνει την διεύθυνση του εσωτερικού καταχωρητή στον οποίο θέλει να γράψει
- Στέλνει το byte δεδομένων
- Στέλνει (προαιρετικά)οποιονδήποτε αριθμό επιπλέον byte
- Στέλνει την ακολουθία λήξης



Σχήμα 1.5.3: Διεύθυνση Slave



Σχήμα 1.5.4: Διεύθυνση Καταχωρητή



Σχήμα 1.5.5: Λήξη μετά την μετάδοση των δεδομένων

❖ Ανάγνωση από την slave συσκευή

Πριν να διαβάσουμε δεδομένα από μία slave συσκευή, πρέπει να την πληροφορήσουμε ποιον εσωτερικό καταχωρητή της θέλουνε να διαβάσουμε. Έτσι μία ανάγνωση από την slave συσκευή στην πραγματικότητα ξεκινά με μία εγγραφή σ' αυτήν. Έτσι η ακολουθία που ακολουθείται για να διαβάσει μία master από μία slave έχει ως εξής:

Η master συσκευή:

- Στέλνει την ακολουθία έναρξης
- Στέλνει την διεύθυνση της slave συσκευής με το R/W bit low (εγγραφή, άρτια διεύθυνση).

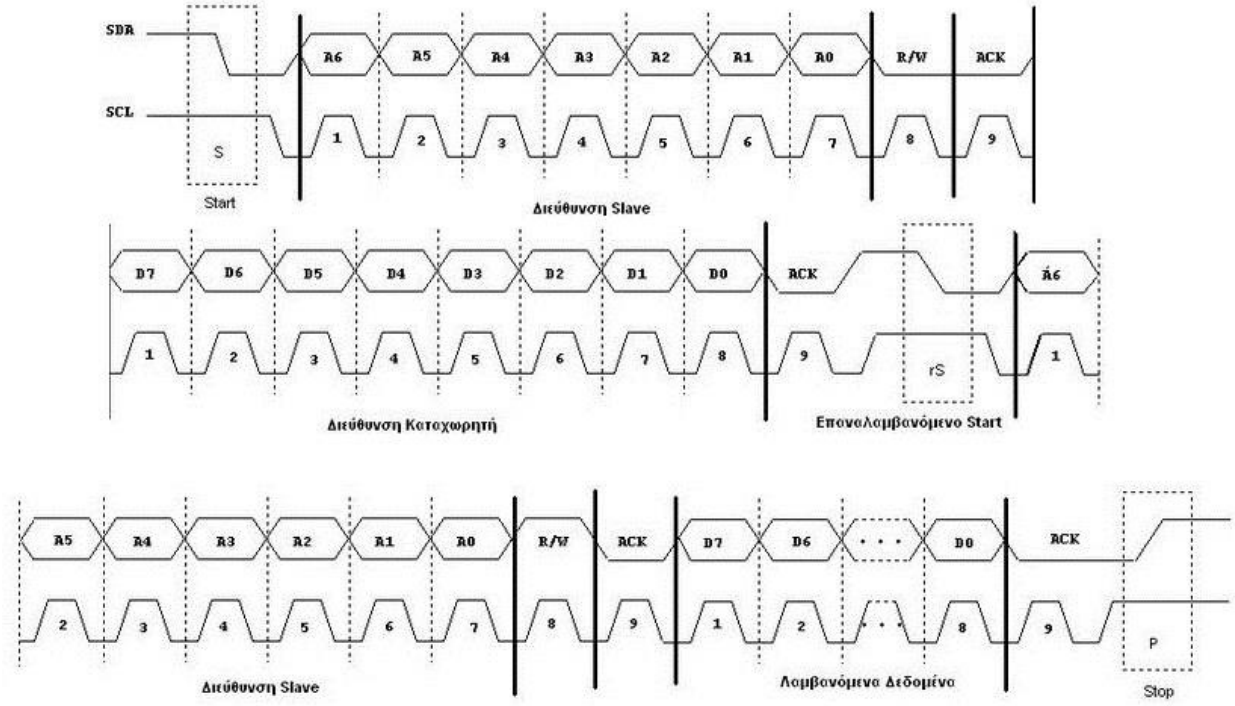
- Στέλνει την διεύθυνση του εσωτερικού καταχωρητή από τον οποίο θέλει να διαβάσει.

Με τα δύο προηγούμενα βήματα, γράφεται ο καταχωρητής-δείκτης της slave συσκευής, σύμφωνα με αυτά που αναφέραμε νωρίτερα για γρήγορη πρόσβαση στα δεδομένα

- Στέλνει πάλι την ακολουθία έναρξης (επαναλαμβανόμενη έναρξη rS)
- Στέλνει πάλι την διεύθυνση της slave συσκευής με το R/W bit high (περιττή διεύθυνση), για να δηλώσει ότι επιθυμεί ανάγνωση

Διαβάζει τα δεδομένα (ένα ή περισσότερα byte). Μετά από την λήψη κάθε byte, η συσκευή που λαμβάνει επιβεβαιώνει (ACK) τη λήψη.

- Στέλνει την ακολουθία λήξης



Σχήμα1.5.6: Ακολουθία Επαναλαμβανόμενης Αποστολής Δεδομένων

❖ **Εφαρμογές**

Ο δίαυλος I²C είναι κατάλληλος για περιφερειακά όπου η απλότητα και το χαμηλό κόστος κατασκευής είναι σημαντικότερα από την ταχύτητα. Συνηθισμένες εφαρμογές του διαύλου είναι:

- Ανάγνωση-εγγραφή σειριακών μνημών EEPROM
- Πρόσβαση σε χαμηλής ταχύτητας μετατροπείς αναλογικού σήματος σε ψηφιακό (ADC) ή ψηφιακού σε αναλογικό (DAC).
- Ανάγνωση αισθητήρων με σύνδεση I²C
- Ανάγνωση ρολογιών πραγματικού χρόνου (Real Time Clocks)
- Ανάγνωση επιτηρητών Hardware και διαγνωστικών αισθητήρων, όπως πχ θερμοστατών CPU και ταχύτητας ανεμιστήρων
- Ενεργοποίηση-Απενεργοποίηση τροφοδοσίας τμημάτων συστημάτων κλπ.

Στην αγορά υπάρχουν μικροελεγκτές που έχουν ενσωματωμένες θύρες I²C, αλλά ένα ιδιαίτερα ισχυρό χαρακτηριστικό του διαύλου I²C, είναι ότι ένας μικροελεγκτής μπορεί να εξομοιώσει τις θύρες I²C μόνο με γενικής χρήσης ακροδέκτες (general purpose I/O) και λογισμικό, χωρίς να χρειάζεται να έχει εξειδικευμένο hardware.

ΚΕΦΑΛΑΙΟ 2

ΥΛΙΚΆ ΚΑΙ ΛΟΓΙΣΜΙΚΟ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

2.1 ΤΟ ΣΥΣΤΗΜΑ XPORT-05R ΓΙΑ ΤΗΝ ΣΥΝΔΕΣΗ ΜΕ ΤΟ ΔΙΑΔΥΚΤΙΟ

Το Xport-05R [5] είναι μία ολοκληρωμένη λύση που επιτρέπει σύνδεση στο διαδίκτυο σε κάθε συσκευή μέσω της συριακής θύρας που διαθέτει. Προσθέτοντας το Xport στο σχεδιασμό ενός προϊόντος, οι κατασκευαστές μειώνουν το μέγεθος έως και 80% προσφέροντας παράλληλα σύνδεση Ethernet σε χρόνους ρεκόρ.

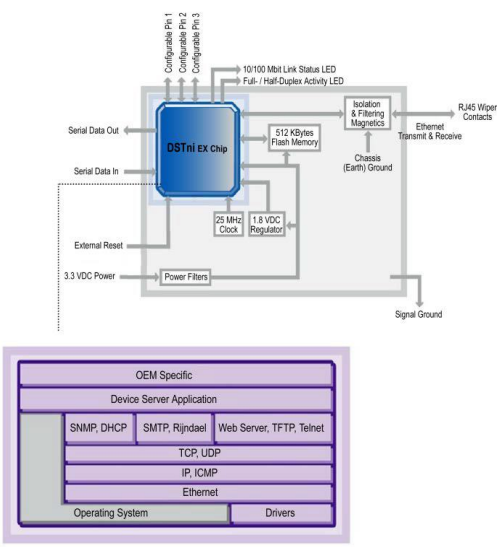
Σχήμα 2.1.1: XPORT-05R

Το Xport προσφέρει το μεγαλύτερο επίπεδο ολοκλήρωσης που διατίθεται σαν μία συσκευή διακομιστή. Σε ένα πακέτο RJ45 υφίσταται ένας ελεγκτής DSTni-EX 186, μνήμη, πομποδέκτης Ethernet 10/100,συριακή θύρα υψηλής ταχύτητας, δύο Led διάγνωσης και 3 προγραμματιζόμενοι ακροδέκτες I/O. Σε χώρο που είναι μικρότερος από το μέγεθος μιας σειριακής θύρας το Xport παρέχει μια ολοκληρωμένη διασύνδεση δικτύου.

Για την ενεργοποίηση πρόσβασης σε ένα τοπικό δίκτυο ή στο διαδίκτυο, το Xport ενσωματώνει ένα πλήρως ανεπτυγμένο πρωτόκολλο TCP/IP δικτύου και OS. Το Xport περιλαμβάνει επίσης ένα ενσωματωμένο server ώστε να ρυθμίζει, παρακολουθεί ή λύνει προβλήματα στη συνδεδεμένη συσκευή.

Κατά την ανάγκη χειροκίνητης διασύνδεσης και χρήσης οικείων εργαλείων, το Xport μπορεί να παρέχει ιστοσελίδες σε προγράμματα περιήγησης. Αποτελεί έναν σύνδεσμο μεταξύ του χρήστη και της συσκευής με το τοπικό δίκτυο ή στο διαδίκτυο.

Το λογισμικό βασισμένο στο λειτουργικό σύστημα των windows που παρέχεται για την ρύθμιση του Xport και ονομάζεται DeviceInstaller μας επιτρέπει μέσω ρυθμίσεων των διαφόρων παραμέτρων, απλοποιεί την εγκατάσταση και τη ρύθμιση της συσκευής. Συνδέεται επίσης μέσω συριακής θύρας, ή μέσω δικτύου χρησιμοποιώντας το Telnet ή κάποιο πρόγραμμα περιήγησης. Η φλας μνήμη παρέχει την δυνατότητα χρήσης και αποθήκευσης ιστοσελίδων χωρίς την άμεση χρήση τροφοδοσίας και επιτρέπει μελλοντικές αναβαθμίσεις.



❖ Περιγραφή Hardware και Software.

Το XPort είναι μια ολοκληρωμένη λύση (hardware και λογισμικού) για σύνδεση με το διαδίκτυο των συσκευών με τις οποίες συνδέεται. Συσκευασμένα σε μια υποδοχή RJ45, αυτή η ισχυρή συσκευή διακομιστή έρχεται μαζί με ένα συνδεσμολογία Ethernet 10BASE-T / 100BASE-TX, αξιόπιστο και δοκιμασμένο λειτουργικό σύστημα που αποθηκεύονται σε μνήμη flash, ένα ενσωματωμένο web server, καθώς και ένα πλήρες TCP / IP πρωτόκολλο, και που βασίζονται σε πρότυπα (AES) κρυπτογράφησης.

Το λογισμικό του Xport τρέχει σε ένα ελεγκτή DSTni-EX, ο οποίος έχει μνήμη 256 KB τύπου SRAM, 16 KB μνήμη έναρξης, και μία διεύθυνση MAC με ολοκληρωμένο 10/100 BASE-TX PHY πρωτόκολλο. Το XPort επικοινωνεί με την τερματική συσκευή μέσω μιας 3,3V σειριακής διεπαφής και τρείς προγραμματιζόμενους I/O ακροδέκτες. Οι 512 KB μνήμες φλάς περιλαμβάνονται για την αποθήκευση ιστοσελίδων. Το XPort λειτουργεί σε 3,3V και έχει ένα ενσωματωμένο εποπτικό κύκλωμα που θα προκαλέσει επαναφορά εάν η τάση

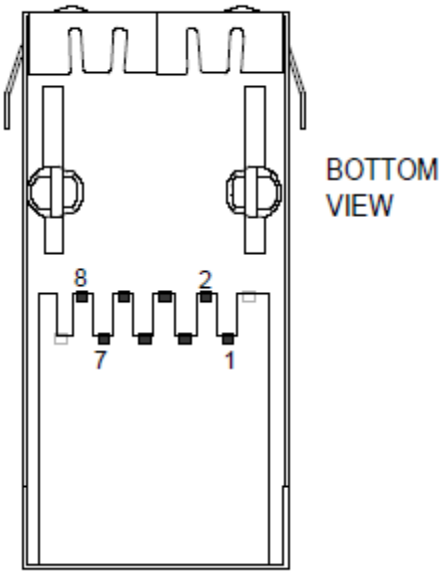
τροφοδοσίας πέσει σε αναξιόπιστα επίπεδα (2,7 V). Ένας ενσωματωμένος ρυθμιστής του 1,8V οδηγεί το πυρήνα επεξεργασίας του ελεγκτή.

Ένα καλώδιο Ethernet RJ45 συνδέεται απευθείας σε ένα XPort. Η Μαγνητική θωράκιση Ethernet από θορύβους καθώς και τα Led διάγνωσης είναι ενσωματωμένα. Το XPort σχεδιάστηκε για να πληροί τα επίπεδα των εκπομπών κατηγορίας Β, γεγονός που καθιστά την ηλεκτρομηχανική ολοκλήρωση πολύ απλή.

- **Οι σειριακές έξοδοι σε επίπεδο PCB**

Οι ακροδέκτες του XPORT-05R σε επίπεδο πλακέτας έχουν την παρακάτω συνδεσμολογία όπως φαίνεται στο Σχήμα 2.**Error! No text of specified style in document.** 1.2 και αντιστοιχούν στις παρακάτω λειτουργίες

Σχήμα 1..5.**Error! No text of specified style in document.**.11 Γείωση
2 *3,3VPowerin
3 Εξωτερική επαναφορά
4 Σειριακή έξοδος δεδομένων
5 Σειριακή είσοδος δεδομένων
6 CP1 Προγραμματιζόμενη είσοδο έξοδο / ΄Εξοδος ελέγχου ροής
7 CP2 Προγραμματιζόμενη είσοδο έξοδο / Έξοδος ελέγχου modem
8 CP3Προγραμματιζόμενη είσοδο έξοδο / Είσοδος ελέγχου ροής/ Είσοδος ελέγχου modem



Σχήμα 2. Error! No text of specified style in document. **1.2 Κάτω όψη XPORT-05R με αρίθμηση των ακροδεκτών του**

- **Σήματα διεπαφής Ethernet**

Σήμα Εισόδου/Εξόδου	Ακροδέκτης	Σήμα Αναφοράς
TX+ Out	1	Transmit Data +
TX- Out	2	Transmit Data –
RX+ In	3	Receive Data +
RX- In	4	Receive Data –
Δεν Χρησιμοποιείται	5	Τερματισμός
Δεν Χρησιμοποιείται	6	Τερματισμός
Δεν Χρησιμοποιείται	7	Τερματισμός
Δεν Χρησιμοποιείται	8	Τερματισμός
Ασπίδα θορύβου		Γείωση Κελύφους

❖ **Ενίσχυση πρωτοκόλλου**

Το XPort χρησιμοποιεί πρωτόκολλο ίντερνετ για την επικοινωνία δικτύων και πρωτόκολλο ελέγχου μεταφοράς ώστε να βεβαιώσει ότι κανένα δεδομένο δε θα χαθεί και οτιδήποτε σταλεί φτάνει σωστά στο στόχο.

❖ **LED**

Περιλαμβάνονται δύο δίχρωμα LED στο μπροστινό μέρος του XPort με τις ακόλουθες χρωματικές ενδείξεις.

Αριστερό LED σύνδεσης		**Δεξή LED δραστηριότητας**	
Χρώμα	**Έννοια**	**Χρώμα**	**Έννοια**
Σβηστό	Χωρίς Σύνδεση	Σβηστό	Χωρίς δραστηριότητα
Κεχριμπάρι	10Mbps	Κεχριμπάρι	Half-Duplex
Πράσινο	100Mbps	Πράσινο	Full-Duplex

2.2 Ο ΕΠΕΞΕΡΓΑΣΤΗΣ DS89C450

❖ Γενική Περιγραφή

Ο επεξεργαστής DS89C450 [6] προσφέρει την υψηλότερη απόδοση που διατίθενται σε μικροελεγκτές συμβατούς με την αρχιτεκτονική του 8051. Διαθέτουν τελευταίου τύπου σχεδιασμένους πυρήνες επεξεργαστών που εκτελούν εντολές 12 φορές ταχύτερα από ότι ο αρχικός 8051 στην ίδια κρυσταλλική ταχύτητα. Τυπικές εφαρμογές έχουν ταχύτητα αυξημένη ακόμη και κατά 10x. Στις 1 εκατομμύριο εντολές το δευτερόλεπτο ανά ένα MHz, οι μικροελεγκτές επιτυγχάνουν 33 MIPS απόδοση στα 33 MHzμέγιστο ρυθμό ρολογιού.

❖ Αναλυτική περιγραφή

Οι DS89C430 και DS89C450 είναι συμβατοί και με τα τρία πακέτα του προτύπου 8051 και περιλαμβάνουν πρότυπες πυγές, όπως τρεις χρονομετρητές, σειριακή θύρα και τέσσερεις 8-bit I/O θύρες. Μεταξύ τους διαφέρουν μόνο κατά το ποσό της εσωτερικής μνήμης 16 kB / 64 kB, το οποίο μπορεί να προγραμματιστεί εσωτερικά στο σύστημα από μια σειριακή θύρα χρησιμοποιώντας ROM το οποίο φορτώνεται με το κατάλληλο λογισμικό. Το φλας μπορεί επίσης να φορτωθεί εξωτερικά με τη χρήση εμπορικών παράλληλων προγραμμάτων.
Οι DS89C430 και DS89C450 περιλαμβάνουν 1 kB μνήμης RAM, μια δεύτερη σειριακή θύρα , επτά επιπλέον διακόπτες, δύο επιπλέον επίπεδα διακοπής προτεραιότητας, προγραμματιζόμενο χρονόμετρο ελέγχου, οθόνη και διακόπτη επαναφοράς. Οι διπλοί δείκτες δεδομένων περιλαμβάνονται για να επιταχύνουν κινήσεις δεδομένων μνήμης με περαιτέρω βελτιώσεις που προέρχονται από τη λειτουργία αυτόματης αύξησης/ μείωσης και εναλλαγής. Η ταχύτητα μνήμης δεδομένων του MOVX μπορεί να προσαρμοστεί με την προσθήκη τιμών έως 10 κύκλων μηχανής για ευελιξία στην επιλεγόμενη εξωτερική μνήμη και περιφερειακά.

Η μονάδα διαχείρισης ενέργειας καταναλώνει σημαντικά χαμηλότερα με τη μείωση του ρυθμού εκτέλεσης του CPU από μια περίοδο κύκλου ρολογιού σε 1024. Ό διακόπτης επαναφοράς μπορεί να ακυρώσει αυτόματα τη διαδικασία για να επιτραπεί η κανονική ταχύτητα. Στις EMI (Electro-Magnetic Interference ή ελληνικά Ηλεκτρομαγνητική παρεμβολή) ευαίσθητες εφαρμογές, ο μικροελεγκτής μπορεί να απενεργοποιήσει το σήμα όταν ο επεξεργαστής δεν έχει πρόσβαση στην εξωτερική μνήμη.

- **Δείκτες δεδομένων**

Οι δείκτες δεδομένων χρησιμοποιούνται για να εκχωρηθεί μια διεύθυνση μνήμης για τις οδηγίες MOVX. Αυτή η διεύθυνση μπορεί να υποδείξει μια θέση MOVXRAMή μια μνήμη που έχει χαρτογραφηθεί ως περιφερειακή. Δύο δείκτες είναι χρήσιμοι κατά τη μετακίνηση των δεδομένων από τη μία περιοχή μνήμης στην άλλη. Ο χρήστης μπορεί να επιλέξει τον ενεργό δείκτη μέσο ενός ειδικού SFRbit ή μπορεί να ενεργοποιήσει την αυτόματη εναλλαγή χαρακτηριστικό για την τροποποίηση της επιλογής δέκτη. Επίσης παρέχει αυτόματη αύξηση ή μείωση της τρέχουσας DPTR.

- **Δείκτης στοίβας**

Ο δείκτης στοίβας υποδηλώνει τη θέση του καταχωρητή στη κορυφή της στοίβας, όπου είναι η τελευταία χρησιμοποιημένη τιμή. Ο χρήστης μπορεί να τοποθετήσει τη στοίβα οπουδήποτε στη RAM θέτοντας το δείκτη στοίβας στην επιθυμητή θέση, αν και τα χαμηλότερα bytesχρησιμοποιούνται συνήθως για τα μητρώα εργασίας.

- **Μετρητές**

Τρείς 16-bitμετρητές είναι διαθέσιμοι. Κάθε μετρητής περιέχεται σε δυο SFR περιοχές και μπορούν να γραφούν ή να αναγνωστούν από το λογισμικό. Οι μετρητές ελέγχονται από άλλα SFR.

- **Συριακές θύρες**

Παρέχονται δύο UARTs τα οποία ελέγχονται και είναι προσβάσιμα από SFRs. Κάθε UART έχει μια διεύθυνση που χρησιμοποιείται για να διαβάσει και να γράψει τη τιμή που περιέχεται στο UART. Η ίδια διεύθυνση χρησιμοποιείται τόσο για την ανάγνωση όσο και την εγγραφή.

- **Οργάνωση μνήμης**

Υπάρχουν τρείς διαφορετικές περιοχές μνήμης του DS89C430, μητρώα, μνήμη προγράμματος και μνήμη δεδομένων. Τα μητρώα βρίσκονται στο τσιπ αλλά οι μνήμες προγράμματος και δεδομένων μπορεί να βρίσκονται είτε μέσε είτε έξω, είτε και μέσα και έξω από το τσιπ. Οι DS89C430/DS89C450 έχουν 16kB/64kB μνήμη προγράμματος στο τσιπ αντίστοιχα, που εφαρμόζεται σε φλας μνήμη και επίσης έχουν 1kB μνήμη δεδομένων στο τσιπ που μπορεί να διαμορφωθεί ως χώρος προγράμματος χρησιμοποιώντας το PRAMEbitστη λειτουργία ROMSIZE. Το DS89C430 χρησιμοποιεί ένα σύστημα που διαχωρίζει τη μνήμη προγραμματισμού από τη μνήμη δεδομένων. Τα τμήματα του προγράμματος και δεδομένων μπορούν να επικαλύπτονται, δεδομένου ότι είναι προσβάσιμες με διαφορετικούς τρόπους. Σε περίπτωση υπέρβασης της μέγιστης διεύθυνση της on-chip μνήμης προγράμματος ή δεδομένων, ο επεξεργαστής DS89C450 εκτελεί μια εξωτερική πρόσβαση στη μνήμη χρησιμοποιώντας τον εκτεταμένο δίαυλο μνήμης. Το σήμα PSEN πηγαίνει σε χαμηλό δυναμικό ώστε να χρησιμεύει ως ένα τσιπ ενεργοποίησης ή ενεργοποίησης εξόδου κατά την εκτέλεση κώδικα από την εξωτερική μνήμη προγράμματος. Οι MOVX οδηγίες ενεργοποιούν το σήμα RD ή

WR για εξωτερική πρόσβαση στη μνήμη δεδομένων MOVX. Η μνήμη ROMSIZE επιτρέπει στο λογισμικό να ρυθμίσετε δυναμικά την μέγιστη διεύθυνση της on-chip μνήμης προγράμματος. Αυτό επιτρέπει στον DS89C430 να ενεργεί ως bootloader για μια εξωτερική μνήμη. Επιτρέπει επίσης τη χρήση της επικάλυψη των εξωτερικών χώρων του προγράμματος. Τα χαμηλότερα 128 bytes της on-chip μνήμης flash ~ αν η ROMSIZE είναι μεγαλύτερη από 0 ~ χρησιμοποιούνται για την αποθήκευση και επαναφορά των φορέων διακοπής (interrupt).Τα 256 bytes της on-chip μνήμης RAM χρησιμεύσουν ως μητρώο καθώς και σαν πρόγραμμα στοίβας, τα οποία διαχωρίζονται από τη μνήμη δεδομένων.

- **Χώρος εγγραφής**

Τα μητρώα που βρίσκονται στα 256 byte της μνήμης RAM του τσιπ μπορούν να χωριστούν σε δύο επιμέρους περιοχές των 128 byteτο κάθε ένα. Τα άνω 128 byte επικαλύπτονται με τα 128 bytetης SFR. Έμμεση δρομολόγηση χρησιμοποιείται για την πρόσβαση των άνω 128 bytetης μνήμης RAM, ενώ ηSRFπεριοχή είναι προσβάσιμη. Υπάρχουν τέσσερις τράπεζες οχτώ λειτουργικών μητρώων στα 128 χαμηλότερα byte της μνήμης RAM. Τα μητρώα είναι γενικού σκοπού θέσεις μνήμης RAM που μπορούν απευθύνονται σε σημεία εντός της επιλεγμένης τράπεζας από οδηγίες που χρησιμοποιούν καταχωρητές R0-R7. Η επιλογή τράπεζας μητρώου ελέγχεται μέσω του καταχωρητή κατάστασης του προγράμματος στην περιοχή της SFR. Τα περιεχόμενα των μητρώων εργασίας μπορεί να χρησιμοποιηθεί για την έμμεση διευθυνσιοδότηση των άνω 128 bytes της επανεγγράψιμης μνήμης RAM . Ατομικά διευθυνσιοδοτούμενα bits στις RAM και SFR περιοχές υποστηρίζουν Boolean λειτουργίες. Στην επανεγγράψιμη περιοχή RAM τα μητρώα 20h-2FH είναι προσπελάσιμη μέσω bit από το λογισμικό χρησιμοποιώντας Boolean οδηγίες λειτουργίας. Μια άλλη χρήση του χώρου επανεγγράψιμης μνήμης RAM είναι για τη στοίβα. Ο δείκτης στοίβας, που περιέχεται στα SFRs, χρησιμοποιείται για την επιλογή των χώρων αποθήκευσης των μεταβλητών του προγράμματος και για τις διευθύνσεις επιστροφής των ενεργειών ελέγχου.

❖ Προγράμματα πρόσβασης μνήμης

Το Πρόγραμμα μνήμης του επεξεργαστή αρχίζει στη διεύθυνση 0000h και εν συνεχεία στη 3FFFh(16Kb) για το DS89C430 και μέσω FFFh(64Kb) για το DS89C450. Κατά την υπέρβαση της μέγιστης εσωτερικής μνήμης του επεξεργαστή, η συσκευή θα προβεί στην χρήση στην εξωτερικής μνήμης. Το μέγιστο της μνήμης του επεξεργαστή αποκωδικοποιείται από την διεύθυνση που επιλέγεται από το λογισμικό, χρησιμοποιώντας το χαρακτηριστικό ROMSIZE. Το λογισμικό μπορεί να προκαλέσει το DS89C430 να συμπεριφέρεται σα μια συσκευή με λιγότερη μνήμη επεξεργαστή. Αυτό είναι ευεργετικό όταν η επικαλυπτόμενος εξωτερική μνήμη χρησιμοποιείται. Το μέγιστο όριο της μνήμης είναι δυναμική μεταβλητή, έτσι, ένα τμήμα της μνήμης μπορεί να αφαιρεθεί (από τον χάρτη μνήμης) και έτσι να υπάρχει πρόσβαση στη εξωτερική μνήμη και στη συνέχεια να αντικατασταθεί με μνήμη επεξεργαστή. Στην πραγματικότητα, όλη η μνήμη επεξεργαστή μπορεί να αφαιρεθεί (από το χάρτη μνήμης)επιτρέποντας στον πλήρη χώρο μνήμης 64kB που πρέπει να αντικατασταθούν από την εξωτερική μνήμη. Οι διευθύνσεις μνήμης του προγράμματος, που είναι μεγαλύτερες από την μέγιστη επιλεγμένη, φορτώνονται αυτόματα από το εξωτερικό του μέρος μέσω των θυρών 0 και 2. Το παρακάτω σχήμα δείχνει μία
απεικόνιση του χάρτη μνήμης.

Το μητρώο ROMSIZE χρησιμοποιείται για να επιλέξουμε το μέγιστο της μνήμης του επεξεργαστή που αποκωδικοποιούνται στην διεύθυνση για τη μνήμη προγράμματος. Για RMS2, RMS1, και RMS0 έχουμε την εξής επίπτωση:

RMS2	RMS1	RMS0	Μέγιστο μνήμης επεξεργαστή προγράμματος (μέγεθος/διεύθυνση)
0	0	0	0kB
0	0	1	1kB/03FFh
0	1	0	2kB/07FFh
0	1	1	4kB/0FFFh
1	0	0	8kB/1FFFh
1	0	1	16kB/3FFFh(DS89C430αυτόματα)
1	1	0	32kB/7FFFh
1	1	1	64kB/FFFFh(DS89C430αυτόματα)

Η επαναφορά της προεπιλεγμένης κατάστασης για όλες τις συσκευές που είναι στο μέγιστο της μνήμης του επεξεργαστή τους προγράμματος. Κατά την πρόσβαση σε εξωτερική μνήμη προγράμματος, το μέγεθος της εξωτερικής μνήμης θα ήταν μη προσπελάσιμο. Για να επιλέξουμε ένα μικρότερο αποτελεσματικό πρόγραμμα το μέγεθος της μνήμης του λογισμικού πρέπει να μεταβληθεί σε bits RMS2 - RMS0. Η μεταβολή αυτών των bits απαιτεί μια χρονομετρημένης πρόσβασης διαδικασία, όπως θα εξηγηθεί αργότερα. Πρέπει να λαμβάνεται προσοχή έτσι ώστε η αλλαγή του μητρώο ROMSIZE δεν διαφθείρει την εκτέλεση του προγράμματος. Για παράδειγμα, εάν υποθέσουμε ότι ένας DS89C430 επεξεργαστής εκτελεί εντολές από τη εσωτερική μνήμη του προγράμματος κοντά στο όριο των 12kB (περίπου 3000 ώρες) και ότι το μητρώο ROMSIZE έχει ρυθμιστεί για ένα 16kB εσωτερικού χώρου πρόγραμμα.

Εάν το λογισμικό ρυθμίζει το μητρώο ROMSIZE να είναι στα 4kB (διευθύνσεις 0000h - 0FFFh) στην τρέχουσα κατάσταση, η συσκευή μεταβαίνει αμέσως στην εξωτερικής μνήμης εκτέλεση του προγράμματος, επειδή ο κώδικας του προγράμματος από 4kB σε 16kB (διευθύνσεις 1000h - 3FFFh) δεν βρίσκεται πλέον στην μνήμη του επεξεργαστή. Αυτό θα μπορούσε να οδηγήσει σε απόκλιση του κώδικα και στην εκτέλεση μιας λανθασμένης εντολής. Η προτεινόμενη

μέθοδος είναι να τροποποιήσει το μητρώο ROMSIZE από μια τοποθεσία στη μνήμη, η οποία είναι εσωτερικά (ή εξωτερικά) τόσο πριν όσο και μετά την λειτουργία. Στο παραπάνω παράδειγμα, η εντολή που τροποποιεί το μητρώο ROMSIZE που πρέπει να βρίσκεται κάτω από το όριο των 4kB (1000h) ή πάνω από το όριο των 16kB (3FFFh), έτσι ώστε να μην επηρεάζεται από την τροποποίηση της μνήμης. Η ίδια προφύλαξη θα πρέπει να εφαρμοστεί εάν το μέγεθος της εσωτερικής μνήμης του προγράμματος έχει τροποποιηθεί κατά την εκτέλεση του από εξωτερική μνήμη του προγράμματος.

Για κενές διεργασίες, η εξωτερική μνήμη είναι προσβάσιμη χρησιμοποιώντας την διεύθυνση πολυπλεξίας / δίαυλος δεδομένων στην P0 και στην MSBδιεύθυνση στην P2. Όσο χρησιμοποιείται ως δίαυλος δεδομένων τα πινάκια δεν είναι 0 και 1 θύρες. Η κατάσταση αυτή ακολουθεί το πρότυπο του 8051, μέθοδος που επεκτείνεται στην μνήμη του επεξεργαστή.

Η εξωτερική πρόσβαση μνήμης του προγράμματος προκύπτει επίσης εάν το πινάκι EA είναι ένα λογικό 0.Το πινάκι EA παρακάμπτει όλες τις ρυθμίσεις του ROMSIZE. Το PSEN σήμα πηγαίνει σε ενεργό (χαμηλή) τιμή για να χρησιμεύσει για να ενεργοποιήσει το τσιπ ή την θύρα εξόδου όταν οι θύρες 0 και 2 ενεργοποιηθούν από την εξωτερική μνήμη του προγράμματος. Τα σήματα RD και WR χρησιμοποιούνται για τον έλεγχο της εξωτερικής μνήμης δεδομένων της συσκευής. Η μνήμη δεδομένων είναι προσβάσιμη από τις οδηγίες της MOVX.

Η εντολή MOVX@Ri χρησιμοποιεί την τιμή του επιλεγμένου καταχωρητή για να παρέχει το λιγότερο σημαντικό ψηφίο (LSB) της διεύθυνσης, ενώ η θήρα 2 παρέχει το περισσότερο σημαντικό ψηφίο(MSB) της διεύθυνσης.

Η εντολή MOVX@Ri χρησιμοποιεί έναν από τους δύο δείκτες δεδομένων για να μεταφέρει δεδομένα σε ολόκληρο τον χώρο της εξωτερικής μνήμης δεδομένων, μεγέθους 64kB. Το λογισμικό επιλέγει το δείκτη δεδομένων που χρησιμοποιείται για την ενεργοποίηση του ψηφίου SEL (DPS.0). Ο DS89C430 παρέχει επίσης μια επιλογή χρήστη για υψηλής ταχύτητας προσπέλαση της εξωτερικής μνήμης με την επαναρίθμηση της διεπαφής της εξωτερικής μνήμης, σε λειτουργία σελίδας (pagemode). Όπου η λειτουργία σελίδας (page mode) ελαχιστοποιεί μερικές από τις εσωτερικές λειτουργίες για συγκεκριμένους τρόπους προσπέλασης.

- **ROMloader**

Το πλήρες πρόγραμμα φλας χωρητικότητας μνήμης στο τσιπ, ασφάλειας μπλοκ φλας και εξωτερικής SRAM μπορεί να προγραμματιστεί στο σύστημα από μια εξωτερική πηγή μέσω της σειριακής θύρας κάτω από τον έλεγχο ενός ενσωματωμένου φορτωτή ROM.

- **Παράλληλη λειτουργία προγραμματισμού**

Ο μικροελεγκτής υποστηρίζει επίσης μια λειτουργία προγραμματισμού, όπως αυτή που χρησιμοποιείται από τις συσκευές προγραμματισμού του εμπορίου. Αυτή η λειτουργία είναι μικρής χρησιμότητας σε κανονικές εφαρμογές.

- **Πηγές διακοπής**

Ο DS89C430 παρέχει 13 πηγές διακοπής (interrupt). Όλες οι διακοπές ελέγχονται από ένα κατά σειρά συνδυασμό επιμέρους απελευθέρωσης bit και μια γενική απελευθέρωσηΕΑ που επιτρέπει τη διακοπή ενεργοποίησης μητρώου.

- **Προτεραιότητα διακοπής**

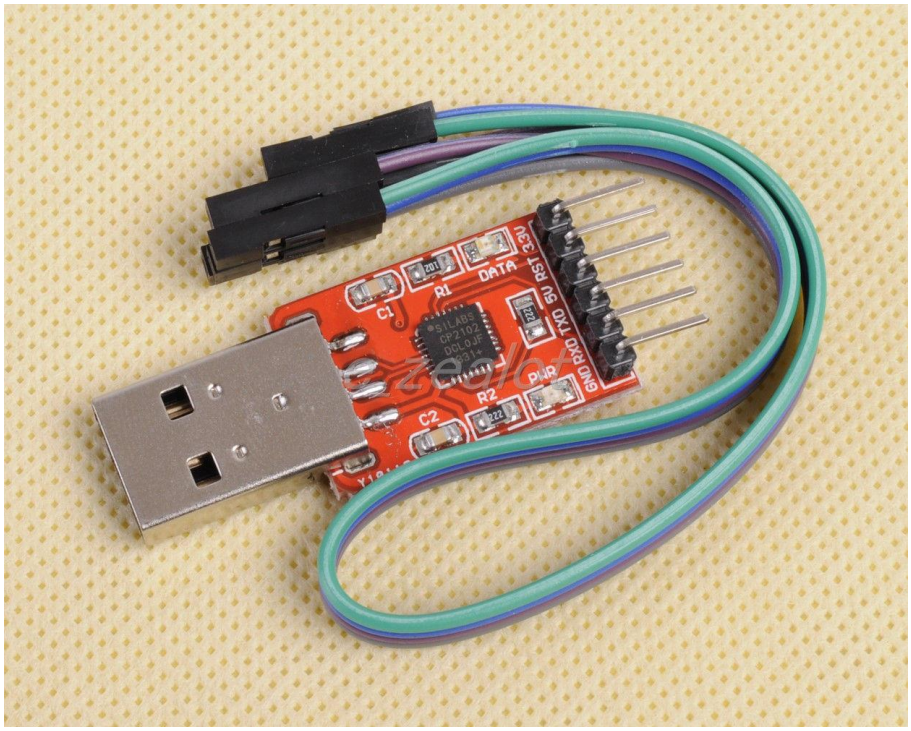
Υπάρχουν πέντε επίπεδα προτεραιότητας διακοπής. Επίπεδο 4-0. Η υψηλότερη προτεραιότητα διακοπής είναι το επίπεδο 4, το οποίο προορίζεται αποκλειστικά για διακοπή power-fail. Όλες οι άλλες διακοπές έχουν μεμονωμένα bitπροτεραιότητας στα μητρώα προτεραιότητας διακοπής. Η διακοπή power-fail έχει πάντα την υψηλότερη προτεραιότητα αν είναι ενεργοποιημένη. Όλες οι διακοπές έχουν μια φυσική ιεραρχία.

- **Μετρητές**

Ο DS89C430 έχει ενσωματωμένα τρία χρονόμετρα 16 bit.Και τα τρία χρονόμετρα μπορούν να χρησιμοποιηθούν είτε ως μετρητέ των εξωτερικών γεγονότων είτε ως μετρητές που μετρούν τους κύκλους.

2.3 Ο ΠΡΟΓΡΑΜΜΑΤΙΣΤΗΣ ΤΟΥ ΕΠΕΞΕΡΓΑΣΤΗ

Για τον προγραμματισμό του επεξεργαστή χρησιμοποιήθηκε ένα σύστημα το οποίο επιτρέπει στον χρήστη να συνδέσει τον υπολογιστή που με το κατάλληλο πρόγραμμα μέσω της θύρας Usb που διαθέτει σειριακά με τις απαιτούμενες θύρες του επεξεργαστή προκειμένου να του φορτώσει το κατάλληλο πρόγραμμα για να εκτελεστούν οι εργασίες που επιθυμεί.

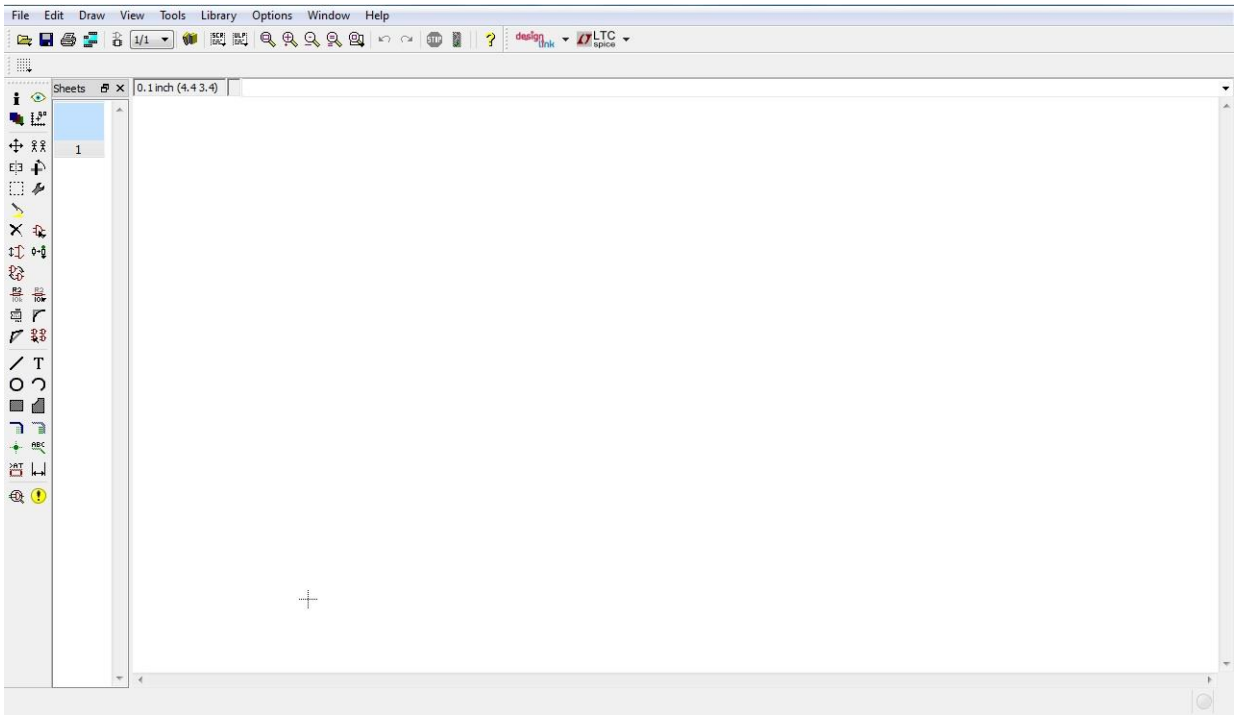


Σχήμα 2.3.1: Το σύστημα σύνδεσης του επεξεργαστή με τον υπολογιστή

Το σύστημα αυτό τοποθετήθηκε ξεχωριστά από την κεντρική σχεδίαση προκειμένου να υπάρχει η δυνατότητα να αξιοποιηθούν οι θύρες σύνδεσης όταν ο επεξεργαστής δεν είναι σε κατάσταση προγραμματισμού.
Σαν σκοπό έχει να στέλνει τις απαραίτητες εντολές σειριακά στον επεξεργαστή και σε συνδυασμό με ένα σύστημα που έχει ενσωματωθεί στην πλακέτα βάζει τον επεξεργαστή σε κατάσταση προγραμματισμού και να τον κρατάει εκεί μέχρι να ολοκληρωθεί ο προγραμματισμός του.

2.4 ΤΟ ΠΡΟΓΡΑΜΜΑ ΣΧΕΔΙΑΣΗΣ ΤΗΣ ΠΛΑΚΕΤΑΣ

Για την σχεδίαση της πλακέτας χρησιμοποιήθηκε το πρόγραμμα EAGLE [7] της CADSoft (http://www.cadsoftusa.com/)

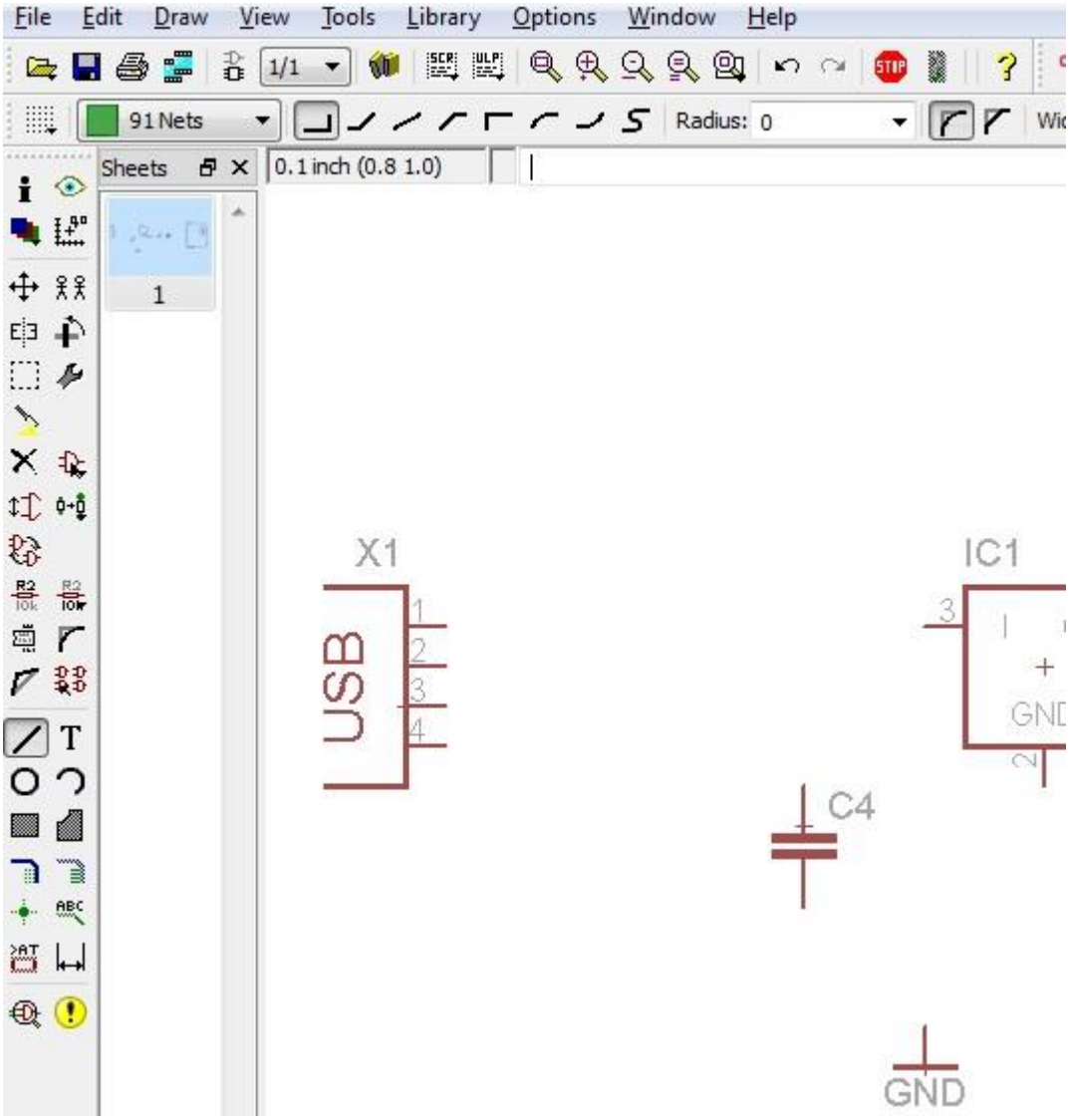


Σχήμα 2.4.1.: Αρχική οθόνη σχεδίασης σχηματικού

Το πρόγραμμα αυτό επιτρέπει στον χρήστη να σχεδιάσει την πλακέτα εύκολα με πληθώρα εξαρτημάτων προς χρήση και προσαρμόζοντας την εύκολα και γρήγορα.

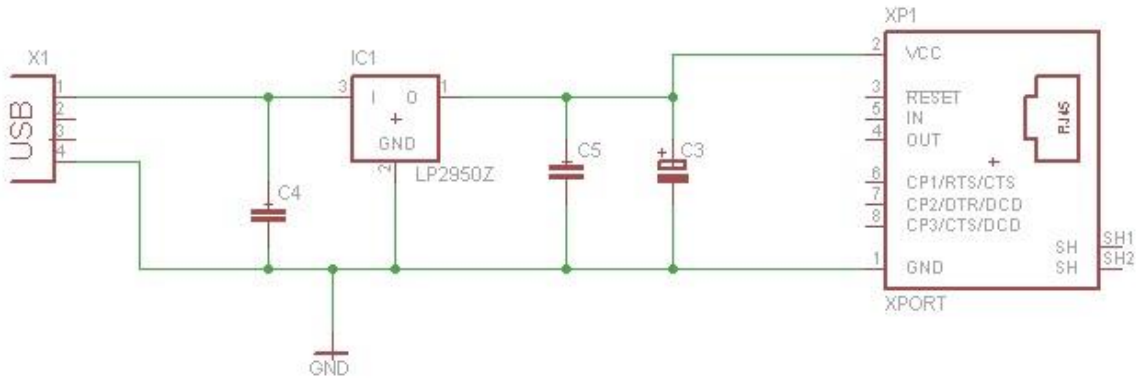
Η σχεδίαση γίνεται σε δύο στάδια και περιλαμβάνει το αρχικό σχηματικό σχεδιάγραμμα και στην απεικόνιση σε επίπεδο πλακέτας. Παρακάτω παρουσιάζεται μια απλή σχεδίαση που περιγράφει μια πλακέτα με την οποία τροφοδοτεί το XPORT με τάση τροφοδοσίας μέσω μίας θύρας USB υπολογιστή προκειμένου να μπορέσει να συνδεθεί και να ρυθμιστεί με τις παραμέτρους που απαιτούνται.

Αφού ανοίξει το πρόγραμμα και δημιουργηθεί καινούργιο project καθώς και καινούργιο σχηματικό σχέδιο ο χρήστης πατάει το πλήκτρο ADD από την εργαλειοθήκη του προγράμματος προκειμένου να προσθέσει τα επιθυμητά εξαρτήματα.



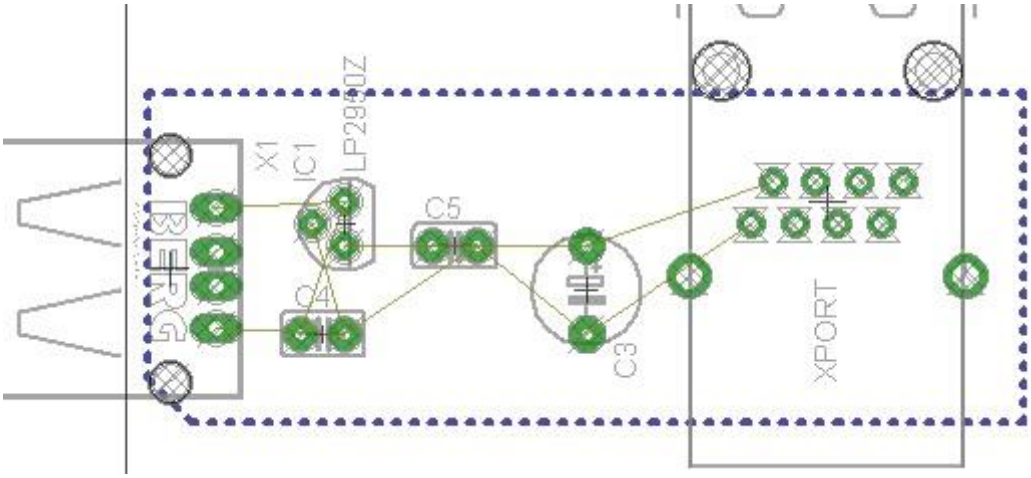
Σχήμα 2.4.2: Προσθήκη των εξαρτημάτων

Εφόσον έχουν προστεθεί όλα τα εξαρτήματα που χρειάζονται, με την βοήθεια του πλήκτρου WIRE στην εργαλειοθήκη του προγράμματος γίνονται όλες οι απαραίτητες συνδέσεις που χρειάζονται συνδέοντας τους εκάστοτε ακροδέκτες με τούς αντίστοιχους που πρέπει να συνδεθούν.



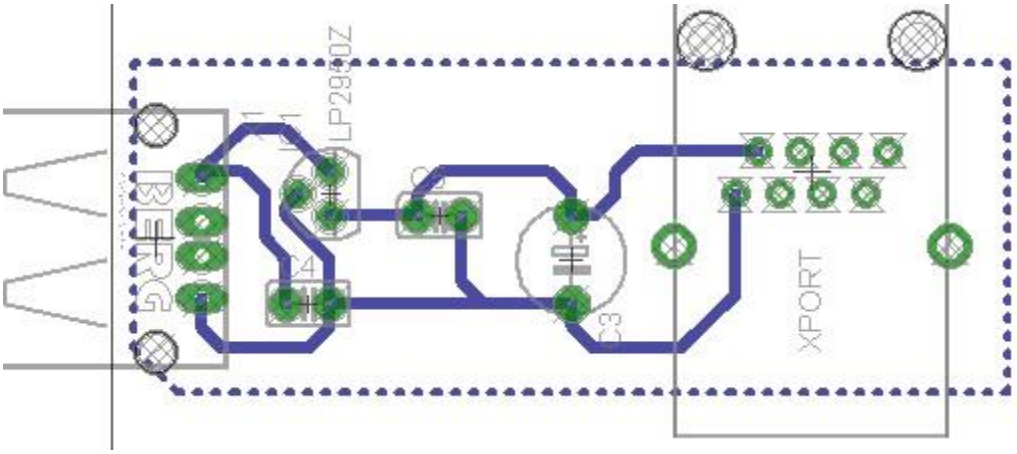
Σχήμα 2.4.3: Σχηματικό τροφοδοσίας XPORT-05R

Έχοντας ολοκληρώσει τις συνδέσεις και πατώντας το εικονίδιο  στην οριζόντια μπάρα εργασίας προκειμένου να μεταφερθούμε σε επίπεδο σχεδιασμού πλακέτας. Εκεί θα δούμε τους ακροδέκτες που αντιστοιχούν στο κάθε εξάρτημα.

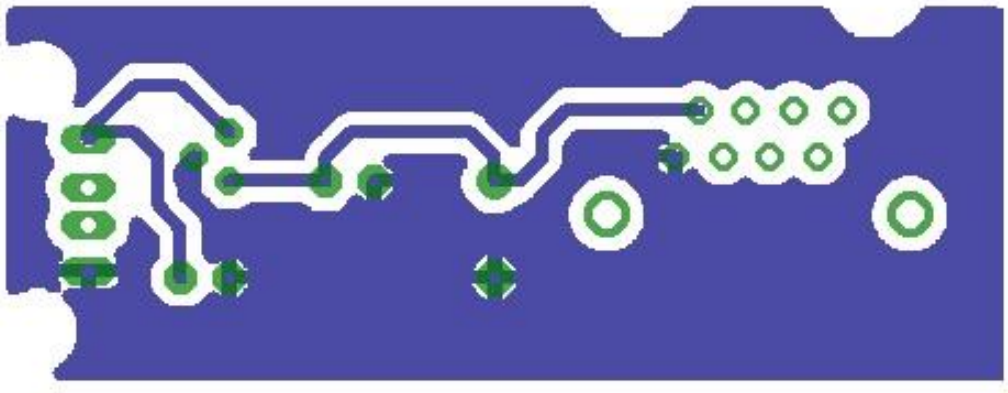


Σχήμα 2.4.4: Σχηματικό πλακέτας χωρίς αγωγούς

Παραπάνω φαίνονται οι συσκευές με τους ακροδέκτες τους και τις συνδέσεις που πρέπει να γίνουν. Στην συνέχεια ο χρήστης πατώντας το πλήκτρο ROUTE συνδέει τους ακροδέκτες προκειμένου να γίνουν οι συνδέσεις. Επιπλέον όταν έχουν ολοκληρωθεί οι συνδέσεις είναι καλό να περιβάλλεται η σχεδίαση μέσα σε ένα αγωγό γείωσης την όλη κατασκευή. Δεν πρέπει να παραλειφθεί να ονομαστεί το πλαίσιο Το τελικό αποτέλεσμα θα έχει ως εξής:



Σχήμα 2.4.5: Σχηματικό με τις συνδέσεις αγωγών



Σχήμα 2.4.6: Τελική μορφή του Τυπωμένου Κυκλώματος έτοιμη προς εμφάνιση

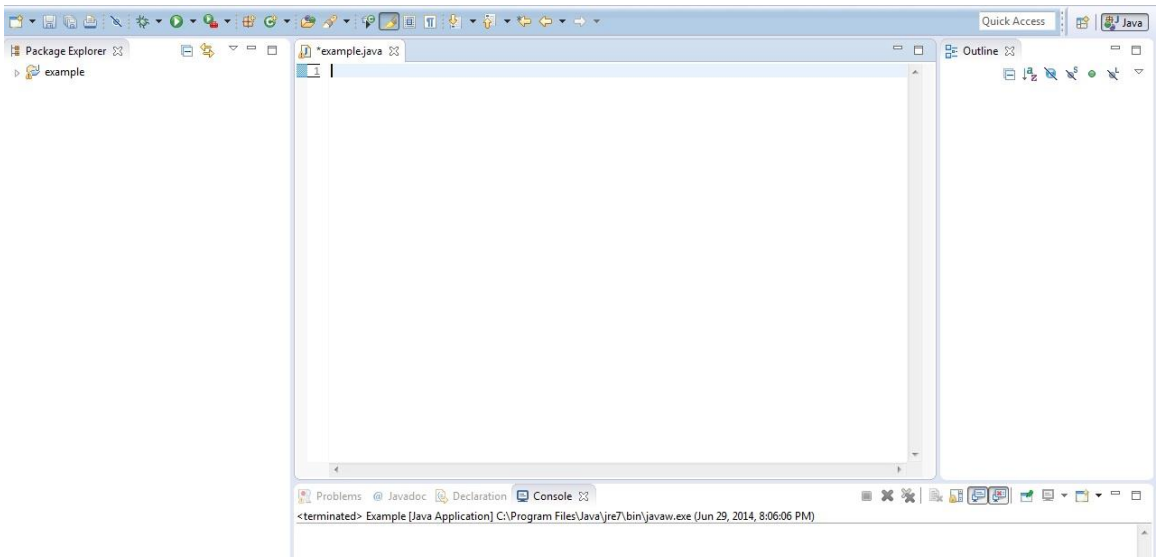
Παρόμοια διαδικασία ακολουθήθηκε και στην δημιουργία και της κυρίως κατασκευής με την μοναδική διαφορά ότι για λόγους εξοικονόμησης χώρου χρησιμοποιήθηκαν δύο επίπεδα σχεδίασης το πρώτο και το τελευταίο για την δημιουργία των αγωγών που απαιτούνταν.

2.5 Ο COMPILER ΤΗΣ ΓΛΩΣΣΑΣ JAVA

Στο κομμάτι του προγραμματισμού της πλακέτας χρειάστηκαν δύο διαφορετικοί μεταφραστές. Ο πρώτος όπως περιγράφεται εδώ και είναι το πρόγραμμα Eclipse [8] της Oracle (http://www.eclipse.org/) και σαν σκοπό έχει τον προγραμματισμό και σχεδιασμό της σελίδας στην γλώσσα JAVA με σκοπό να φορτωθεί στο XPORT για να επιτρέψει την σύνδεση μέσω διαδικτύου και στην συνέχεια με την

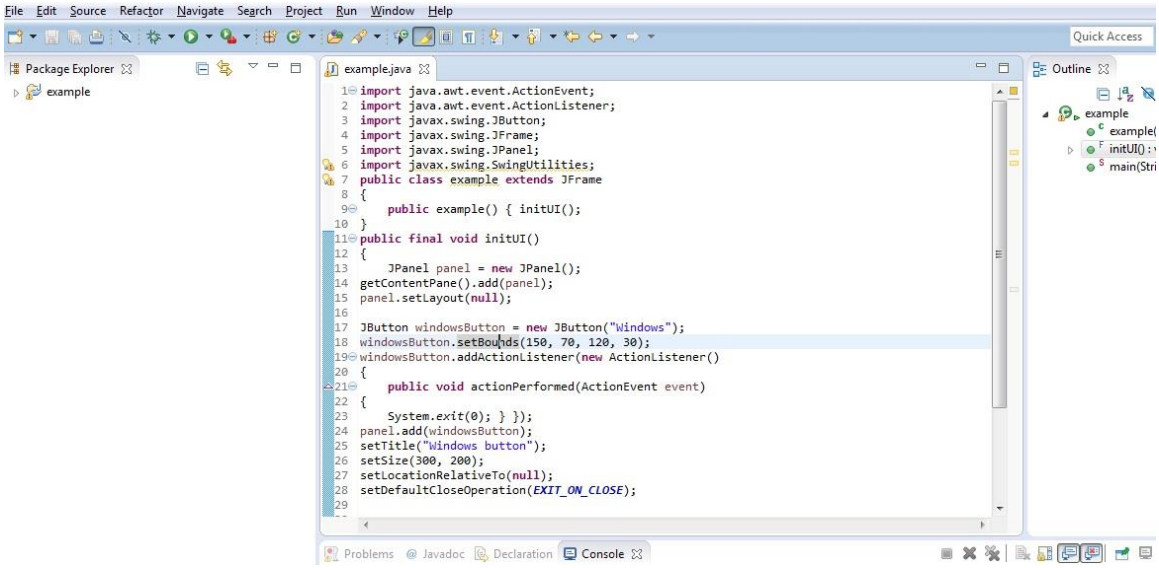
σειρά της να στείλει τις απαραίτητες εντολές στην κεντρική πλακέτα. Ο κώδικας αν και γραμμένος στην γλώσσα JAVA ήταν αρκετά εύκολος στην κατανόηση με βάση τις γνώσεις που παρείχε το ΤΕΙ, δεδομένου ότι έχει πάρα πολλά κοινά σημεία σύνταξης με την γλώσσα C που διδάσκεται κατά την διάρκεια των σπουδών στο ίδρυμα.

Η αρχική εικόνα του προγράμματος μετά την δημιουργία ενός κενού project ονόματι example.java



Σχήμα 2.5.1: Αρχική οθόνη προγράμματος

Στην συνέχεια θα παρατεθεί ένας υποτυπώδης κώδικας ο οποίος δημιουργεί ένα παράθυρο στο οποίο περιέχεται ένα κουμπί με την ονομασία «Windows». το πρόγραμμα όπως φαίνεται παρακάτω μετά την προσθήκη του κώδικα έχει ως εξής.




```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
public class example extends JFrame
{
    public example() { initUI();
}
public final void initUI()
{
    JPanel panel = new JPanel();
getContentPane().add(panel);
panel.setLayout(null);

JButton windowsButton = new JButton("Windows");
windowsButton.setBounds(150, 70, 120, 30);
windowsButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent event)
{
    System.exit(0); } });
panel.add(windowsButton);
setTitle("Windows button");
setSize(300, 200);
setLocationRelativeTo(null);
setDefaultCloseOperation(EXIT_ON_CLOSE);
```

Σχήμα 2.5.2: Σύνταξη του κώδικα στο βασικό παράθυρο

Ο κώδικας με τις απαραίτητες διευκρινήσεις παρατάσσεται παρακάτω:

Ξεκινώντας με την προσθήκη των απαραίτητων βιβλιοθηκών που απαιτούνται προκειμένου να υλοποιηθεί το πρόγραμμα.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
```

Στην συνέχεια προστίθεται ο απαραίτητος κώδικας για να εμφανιστεί το παράθυρο στο οποίο θα τοποθετηθούν τα απαραίτητα στοιχεία.

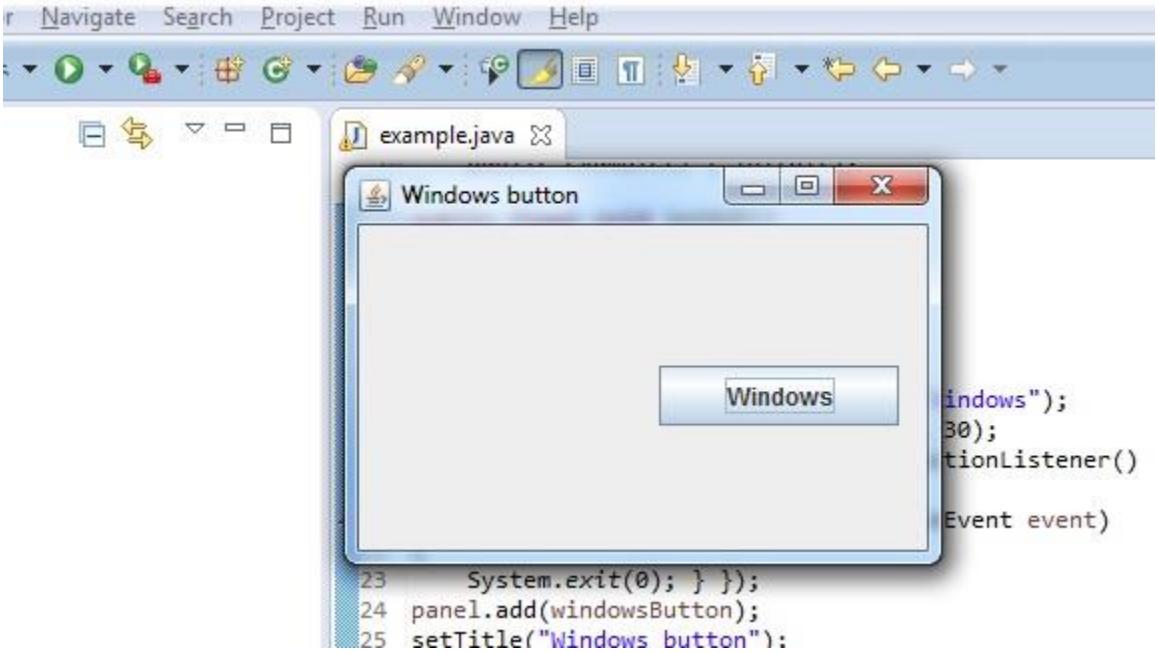
```
public class example extends JFrame
{
    public example() { initUI();
}
public final void initUI()
{
    JPanel panel = new JPanel();
getContentPane().add(panel);
panel.setLayout(null);
```

Το επόμενο κομμάτι κώδικα περιλαμβάνει κατά σειρά τα στοιχεία που αποτελούν το κουμπί που επιθυμεί ο χρήστης να προστεθούν συμπεριλαμβανομένης και την ονομασίας του, της θέσης του καθώς και το μέγεθος που έχει. Η τελευταία συνθήκη στο συγκεκριμένο κομμάτι του προγράμματος έχει σαν σκοπό να πει στο πρόγραμμα τι πρέπει να κάνει σε περίπτωση που πατηθεί το κάθε κουμπί, στην συγκεκριμένη περίπτωση να κλείσει το παράθυρο διαλόγου.

```
JButton windowsButton = new JButton("Windows");
windowsButton.setBounds(150, 70, 120, 30);
windowsButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent event)
{
    System.exit(0); } });
panel.add(windowsButton);
setTitle("Windows button");
setSize(300, 200);
setLocationRelativeTo(null);
setDefaultCloseOperation(EXIT_ON_CLOSE);
}
```

Στο τέλος δηλώνεται εάν θέλει ο χρήστης να εμφανίζεται ή όχι το παράθυρο μετά την ολοκλήρωση του προγράμματος.

```
public static void main(String[] args)
{
    example ex = new example(); ex.setVisible(true);
    }
    }
```



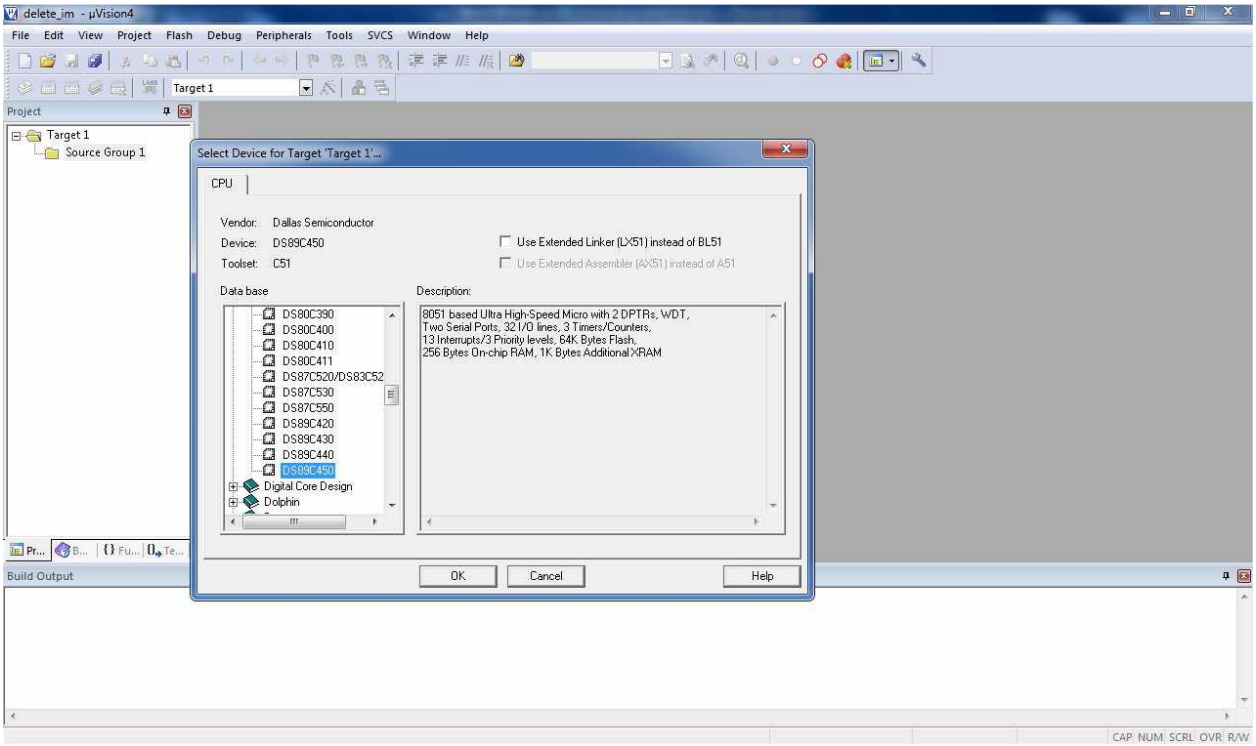
Σχήμα 2.5.3: Αποτέλεσμα Υποτυπώδους κώδικα στην γλώσσα JAVA

Μετά την ολοκλήρωση του προγράμματος πατώντας στο κουμπί Run Example που αντιστοιχίζεται με το σήμα  στην εργαλειοθήκη του προγράμματος εμφανίζεται το αποτέλεσμα του προγραμματισμού που υλοποιήθηκε παραπάνω.

Εφόσον ο κώδικας λειτουργεί κανονικά προκειμένου να φορτωθεί στο XPORT για να λειτουργήσει θα πρέπει να γίνει χρήση άλλου ενός προγράμματος που παρέχεται από την Latronix και ονομάζεται Web2Cob και έχει σαν σκοπό να μετατρέψει τα αρχεία που δημιουργεί το πρόγραμμα Eclipse και είναι τύπου .java σε αρχεία τύπου .cob που είναι τα αρχεία που έχει την δυνατότητα να επεξεργάζεται το XPORT προκειμένου να λειτουργήσει.

2.6 Ο COMPILER ΤΟΥ ΕΠΕΞΕΡΓΑΣΤΗ

Ο κώδικας του μικροελεγκτή αναπτύχθηκε στο περιβάλλον KEIL 4 [9] της εταιρίας uVision (www.keil.com). Πρόκειται ένα πλήρες περιβάλλον προγραμματισμού το οποίο εκτός τον επεξεργαστή κειμένου και τον compiler – assembler, προσφέρει και έναν δυνατό simulator για την αποσφαλμάτωση του κώδικα. Για τη δημιουργία ενός νέου Project, ο χρήστης επιλέγει Project _ New μVision Project, δίνει το όνομα του νέου project και μετά επιλέγει το μικροελεγκτή με τον οποίον θα ασχοληθεί ώστε το πρόγραμμα να γνωρίζει ποίες βιβλιοθήκες να φορτώσει αυτόματα, τα όρια του μικροελεγκτή (μέγιστη συχνότητα κρυστάλλου, μέγεθος RAM, μέγεθος ROM, κλπ) και πιθανών για να ρυθμίσει κάποιες παραμέτρους στον compiler και τον assembler. Οι βιβλιοθήκες περιέχουν (συνδέουν) τα ονόματα των καταχωριστών με τις ανάλογες διευθύνσεις στη μνήμη.

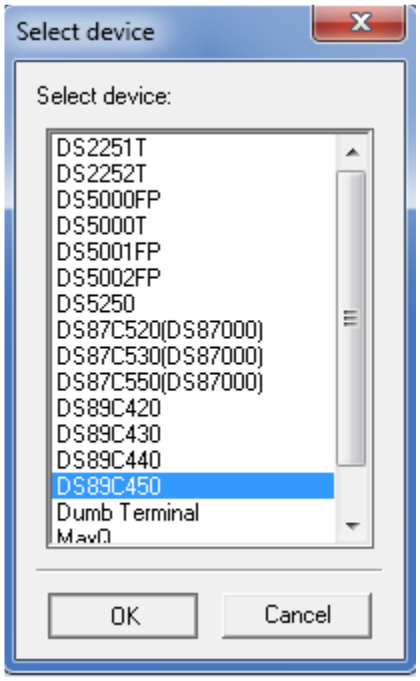


Σχήμα 2.6.1: Επιλογή μικροελεγκτών στο περιβάλλον KEIL4

Αργότερα ο χρήστης μπορεί να φορτώσει στο Project το αρχείο που περιέχει τον κώδικα ή να δημιουργήσει ένα καινούριο και κατόπιν να το φορτώσει, κάνοντας διπλό κλικ στο φάκελο Source Group 1 που βρίσκεται στα αριστερά του προγράμματος, στην καρτέλα Project.

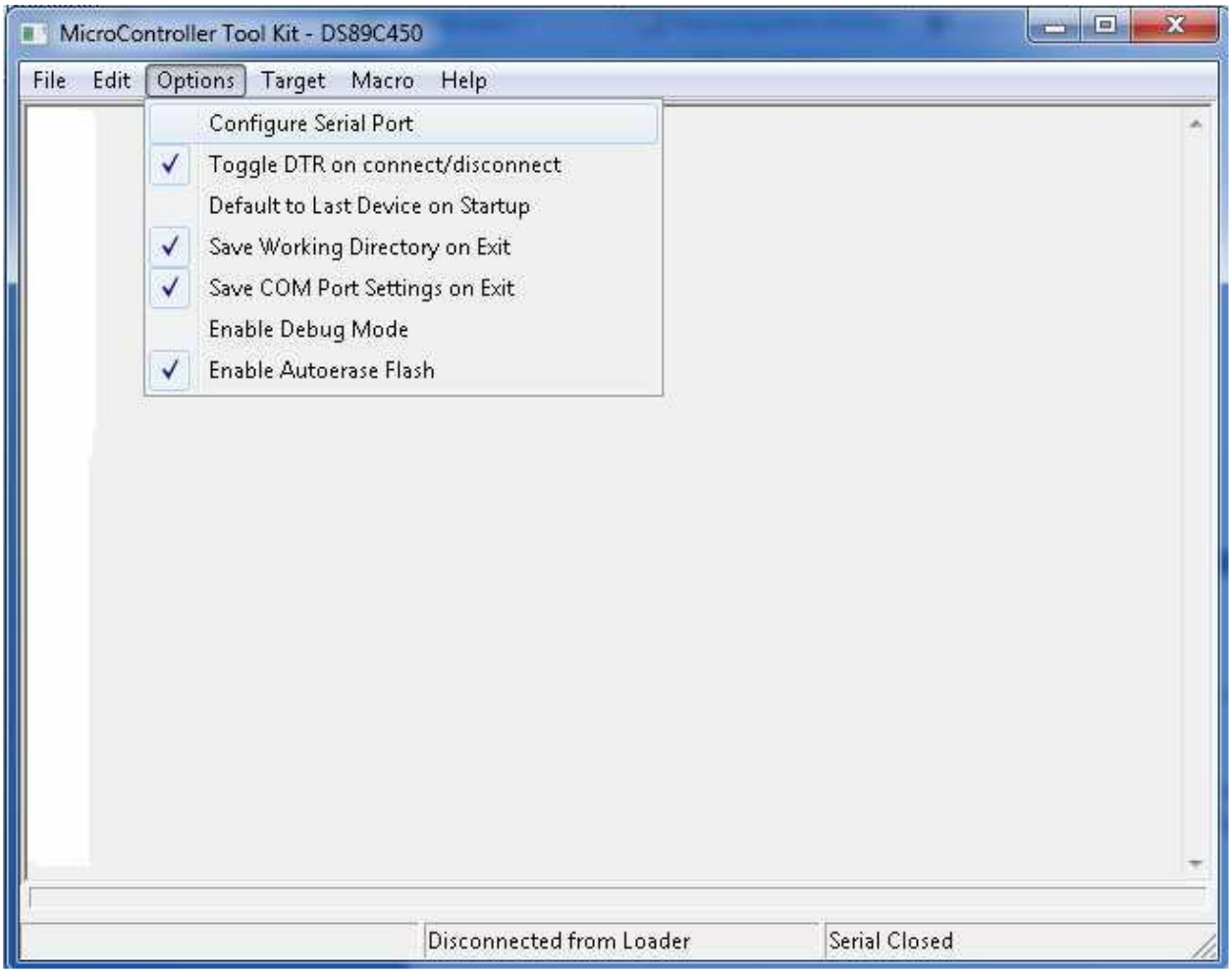
Για να έρθει ο μικροελεγκτής σε κατάσταση προγραμματισμού πρέπει τρεις ακροδέκτες του να έρθουν σε συγκεκριμένη λογική κατάσταση. Πιο συγκεκριμένα, ο ακροδέκτης Reset πρέπει να βρίσκεται σε λογικό '1', ενώ οι ακροδέκτες EA και PSEN σε λογικό '0'. Αυτό επιτυγχάνεται με μια λογική διακοπτών (τρανζίστορ, FET) οι οποίοι οδηγούνται από το σήμα DTR της σειριακής.

Η φόρτωση του προγράμματος στον μικροελεγκτή γίνεται σειριακά και για αυτό το σκοπό χρησιμοποιήθηκε το λογισμικό MTK2 της εταιρίας Maxim (www.maxim-ic.com). Πρόκειται για ένα τερματικό της σειριακής όπως το γνωστό HyperTerminal, με κάποιες παραπάνω λειτουργίες, όπως το κλείδωμα του εκτελέσιμου κώδικα για αποτροπή της ανάγνωσης του, κάποιες βασικές λειτουργίες αποσφαλμάτωσης. Ο προγραμματισμός γίνεται ως εξής μετά την κλήση του προγράμματος: ο χρήστης διαλέγει ποιό μικροελεγκτή επιθυμεί να προγραμματίσει από το παράθυρο που εμφανίζεται στην οθόνη όπως παρακάτω:

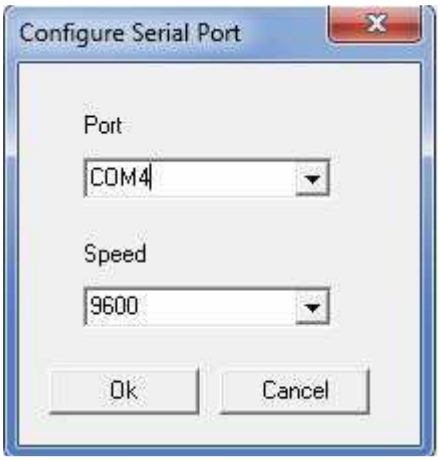


Σχήμα 2.6.2: Παράθυρο επιλογής μικροελεγκτή

Εμφανίζεται η κονσόλα του λογισμικού, ο χρήστης μπορεί να κάνει τις αρχικές ρυθμίσεις για την εκκίνηση της σειριακής στο κατάλληλο baud rate και να συνδεθεί με τον Loader του μικροελεγκτή από το μενού Options > Configure Serial Port



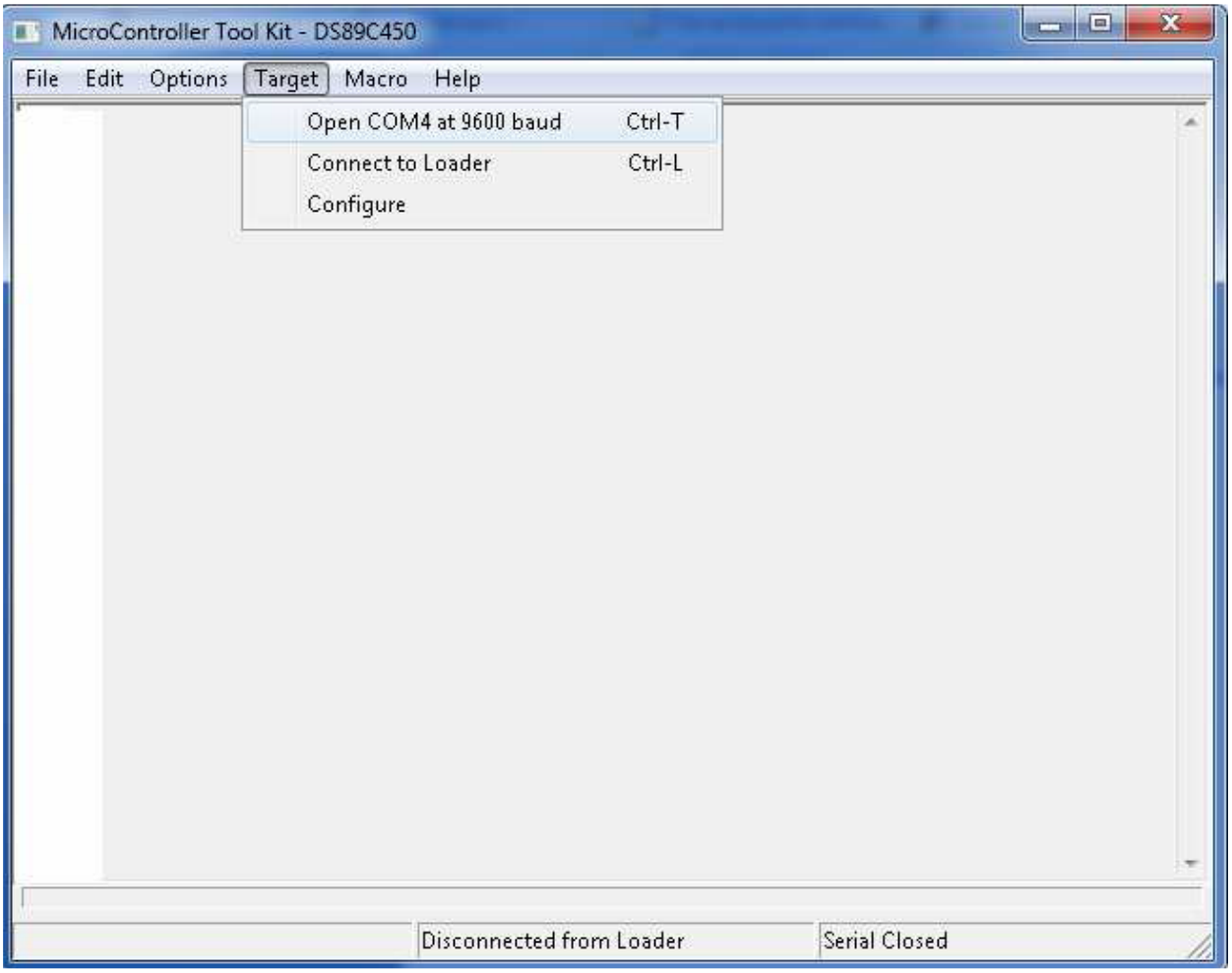
Σχήμα 2.6.3: Παραμετροποίηση της σύνδεσης



Σχήμα 2.6.4: Σύνδεση μέσω σειριακής πόρτας σε συγκεκριμένο BAUD RATE

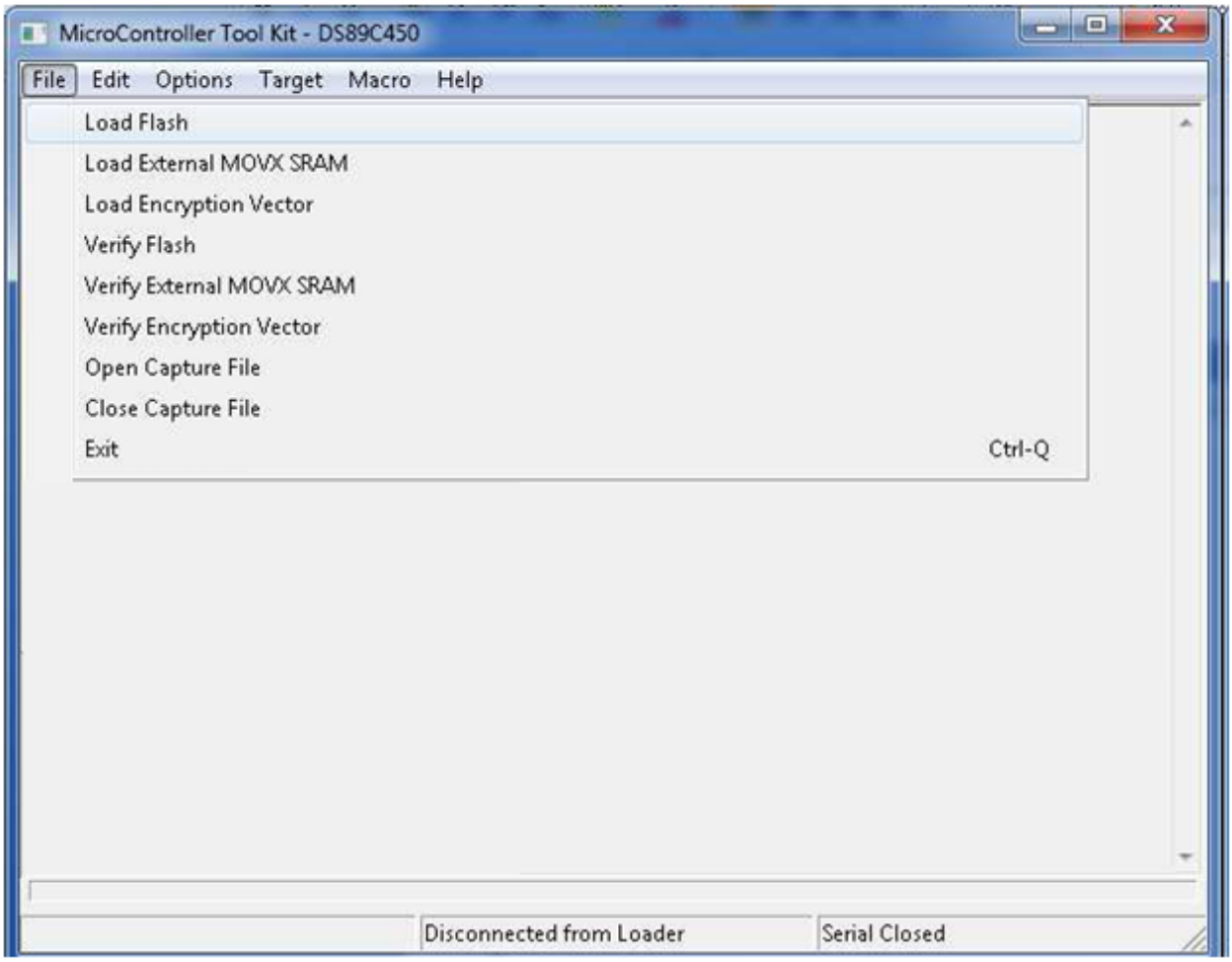
Αργότερα από το μενού Target μπορεί να ανοίξει τη σειριακή κάνοντας κλικ στην επιλογή Open COMx at xxxx baudrate, και αργότερα πάλι από το ίδιο μενού

επιλέγοντας το Connect to Loader, να επικοινωνήσει με τον ενσωματωμένο στον μικροελεγκτή, boot loader του.



Σχήμα 2.6.5: Έναρξης σύνδεσης με τον μικροελεγκτή

Τώρα πλέον είναι δυνατό το κατέβασμα του εκτελέσιμου (.hex) αρχείου προς τον μικροελεγκτή, επιλέγοντας File > Load Flash και δίνοντας τη διαδρομή του εκτελέσιμου αρχείου ως επιλογή.



Σχήμα 2.6.6: Παράθυρο φόρτωσης προγράμματος στον μικροελεγκτή

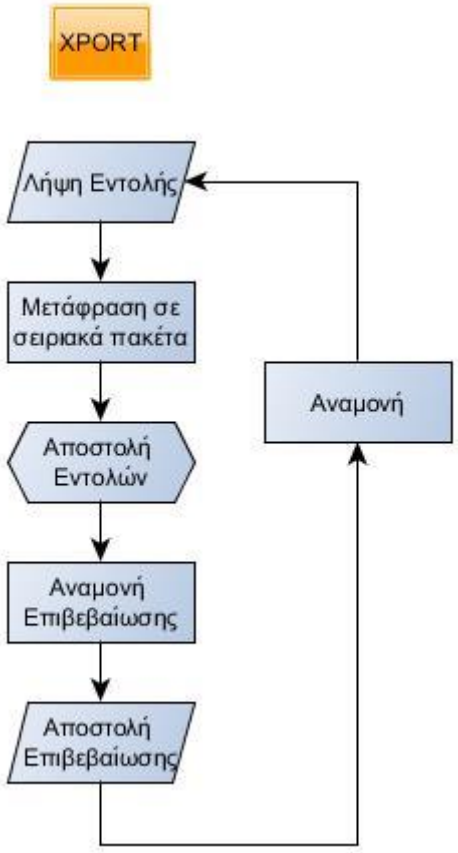
ΚΕΦΑΛΑΙΟ 3

ΜΕΘΟΔΟΛΟΓΙΑ / ΣΧΕΔΙΑΣΗ

3.1 ΑΝΑΛΥΣΗ ΤΩΝ ΔΙΑΔΙΚΑΣΙΩΝ ΠΟΥ ΘΑ ΕΚΤΕΛΕΣΕΙ ΤΟ XPORT

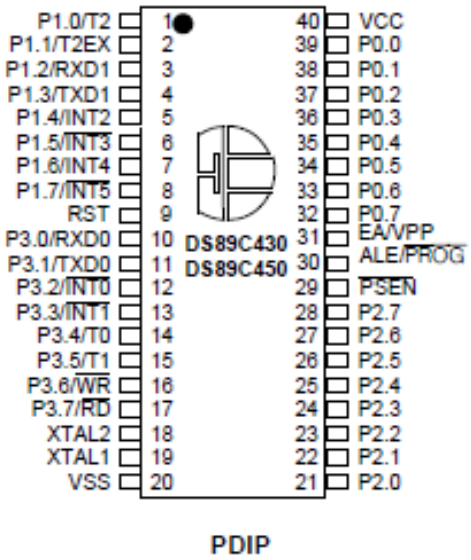
Το XPORT-05R έχει σαν σκοπό να συνδέσει την σχεδίαση μας με το διαδίκτυο. Σε συνδυασμό με τον κώδικα JAVA που θα φορτωθεί στην μνήμη του ο οποίος θα υλοποιεί μία διεπαφή χρήστη μέσω της οποίας ο χρήστης θα έχει πρόσβαση στην πλακέτα για έλεγχο και αλλαγή των εκάστωτε παραμέτρων που επιθυμεί. Στην ουσία το XPORT μέσω της ιστοσελίδας που φιλοξενεί μετατρέπει το πάτημα των πλήκτρων σε σήματα τα οποία στέλνονται στην σειριακή και μέσω αυτής στον κεντρικό επεξεργαστή.

Η διαδικασία αυτή εκτελείται και αντίστροφα όταν το XPORT ζητάει να λάβει ενημέρωση ως προς την κατάσταση των εξόδων την οποία διαβάζει ο επεξεργαστής και τότε το XPORT λαμβάνει από την σειριακή του θύρα τα σήματα από τον επεξεργαστή και τα μετατρέπει με την σειρά του μέσω του κώδικα που εμπεριέχει σε ευδιάκριτα σήματα που εμφανίζονται στην οθόνη διεπαφής χρήστη προκειμένου να μπορούν να ενημερώσουν για το εάν υπάρχει σφάλμα στις εξόδους του επεξεργαστή.



3.2 ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΑΚΡΟΔΕΚΤΩΝ ΕΠΕΞΕΡΓΑΣΤΗ

Στην συνέχεια θα εξηγηθεί η συνδεσμολογία που έχουν οι ακροδέκτες του επεξεργαστή και οι διαδικασίες που εκτελούν.

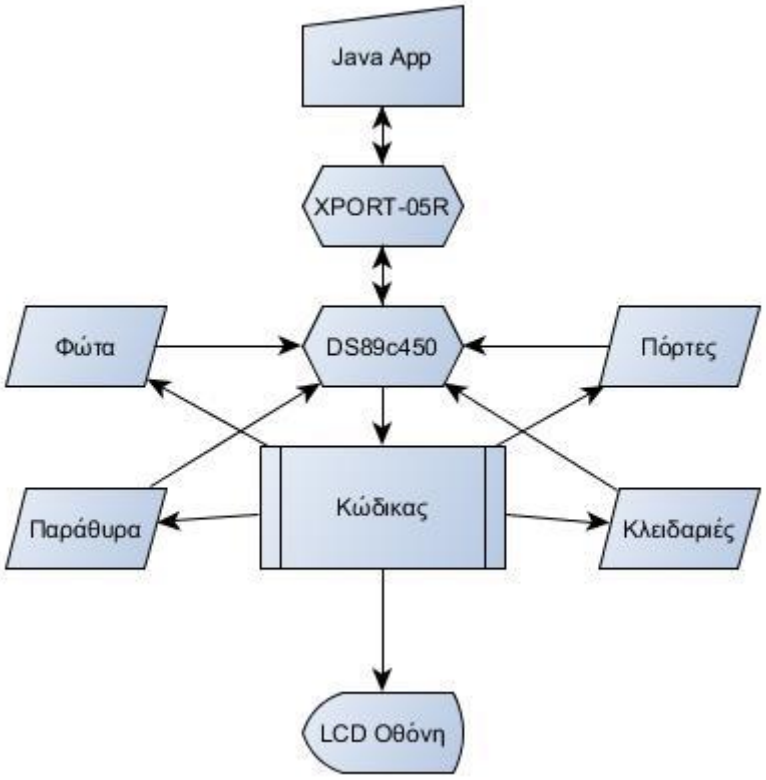


PDIP

Ξεκινώντας από τους πιο βασικούς ακροδέκτες υπαριθμόν 20 και 40 οι οποίοι αντιστοιχούν στην γείωση και τροφοδοσία του επεξεργαστή αντίστοιχα. Για τον προγραμματισμό του έχει χρησιμοποιηθεί η θύρα 9 που αντιστοιχεί στο Reset του επεξεργαστή καθώς και τους ακροδέκτες 29 και 31 που είναι οι ακροδέκτες που λαμβάνουν τα σήματα προγραμματισμού προκειμένου να φορτωθεί το πρόγραμμα στον μικροελεγκτή. Για τον χρονισμό του επεξεργαστή χρησιμοποιήθηκαν οι θύρες 18 και 19 όπου σε συνδυασμό με έναν κρύσταλλο αλλά και δύο πυκνωτές παράγουν την συχνότητα την οποία θα λειτουργήσει ο επεξεργαστής.

Για την σειριακή σύνδεση με το XPORT χρησιμοποιήθηκαν οι ακροδέκτες 3 και 4 που αντιστοιχούν στα RXD και TXD του επεξεργαστή για τις λειτουργίες (R)ead και (T)ransfer που είναι η ανάγνωση και μεταφορά δεδομένων. Τα συγκεκριμένα σήματα συνδέονται ανεστραμμένα στις αντίστοιχες θύρες του XPORT που αντιστοιχούν στα TXD και RXD αντίστοιχα προκειμένου η αποστολή αρχείων του XPORT να συμπίπτει με τον ακροδέκτη ανάγνωσης του επεξεργαστή και το αντίστροφο.

Οι ακροδέκτες 32 μέχρι 36 έχουν συνδεθεί με την οθόνη LCD που περιλαμβάνεται στην σχεδίαση και σκοπό έχουν με την βοήθεια του κώδικα που περιλαμβάνεται στον επεξεργαστή να ενημερώνουν τον χρήστη μέσω κειμένου για το κατά πόσο η εκάστωτε λειτουργία που επιθυμεί πραγματοποιήθηκε.



Οι ακροδέκτες 1 με 2 , 4 με 8 , 10 με 17 και 21 με 28 έχουν ρυθμιστεί προκειμένου να συνδεθούν με τους αισθητήρες για την αποστολή και λήψη δεδομένων από αυτούς, την ανάγνωση των δεδομένων που λαμβάνουν από το περιβάλλον και μετατροπή τους σε κώδικα που αναγνωρίζει ο επεξεργαστής με σκοπό την περεταίρω επεξεργασία τους.
Εν τέλει οι ακροδέκτες 38 και 39 έχουν ρυθμιστεί προκειμένου να υλοποιήσουν το πρωτόκολλο I^2C που αν και δεν χρησιμοποιήθηκε στην λειτουργία της συγκεκριμένης σχεδίασης εξηγείται διεξοδικά σε επίπεδο κώδικα αλλά και αρχής λειτουργίας με σκοπό να μπορεί να συμπεριληφθεί σε μελλοντικές επεκτάσεις της σχεδίασης για την δυνατότητα ελέγχου πολύ μεγαλύτερου πλήθους συσκευών από αυτόν που υλοποιεί η συγκεκριμένη σχεδίαση. Συνοψίζοντας τον κώδικα του επεξεργαστή προκύπτει το επόμενο διάγραμμα ροής που επεξηγεί την σειρά εκτέλεσης των εντολών που λαμβάνουμε καθώς και τις διεργασίες που πραγματοποιούνται έπειτα από κάθε βήμα της όλης διαδικασίας.

3.3 Η ΡΥΘΜΙΣΗ ΤΟΥ XPORT-05R

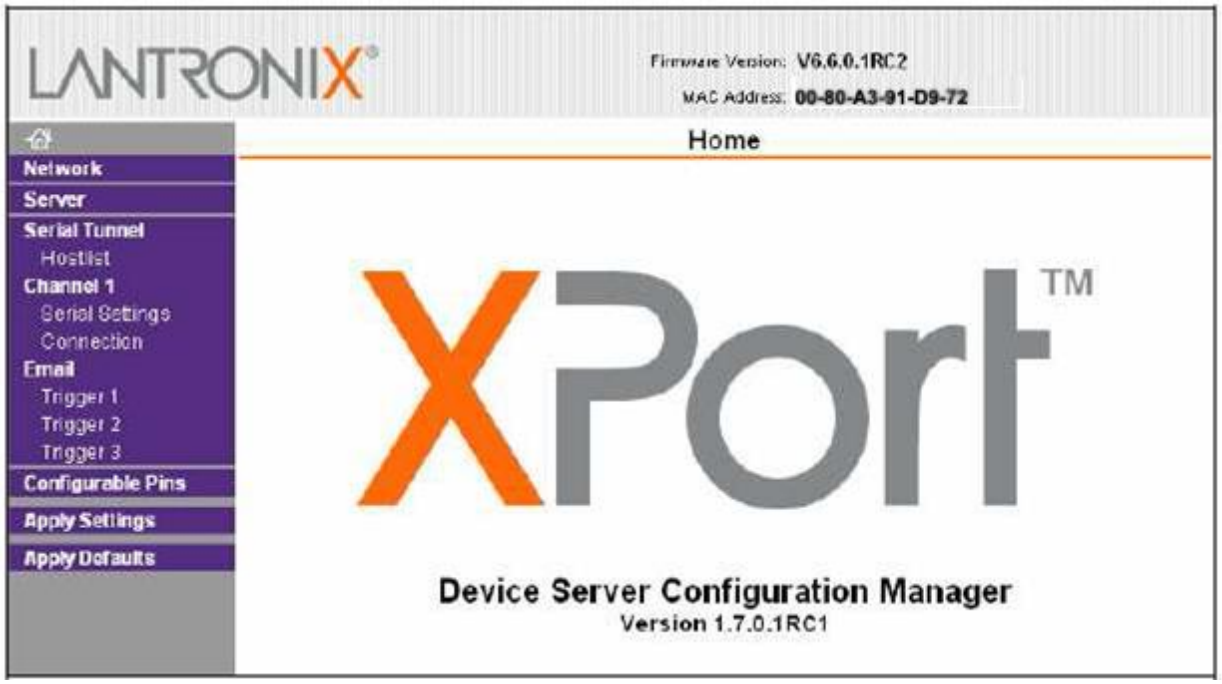
Προτού ο χρήστης ξεκινήσει τη διαδικασία προγραμματισμού της συσκευής, πρέπει να ρυθμίσει κάποιες περεταίρω παραμέτρους όπως, τον τρόπο με τον οποίο η συσκευή θα ανταποκρίνεται στα σειριακά και τα δικτυακά δεδομένα, πως θα διαχειρίζεται τα πακέτα πληροφορίας τα οποία θα λαμβάνει από τη σειριακή, και το πότε να εκκινεί ή να λήγει μια σειριακή συνεδρία. Αυτές η ρυθμίσεις αποθηκεύονται σε μια μη πτητική μνήμη και διατηρούνται χωρίς κάποια τροφοδοσία στο κύκλωμα. Βέβαια, οι ρυθμίσεις αυτές μπορούν να μεταβληθούν οποιαδήποτε στιγμή, η συσκευή θα εφαρμόσει τις νέες ρυθμίσεις αμέσως
μετά την επανεκκίνηση της, η οποία θα γίνει αυτόματα μετά την αποθήκευση αυτών των ρυθμίσεων.
Στο τρέχων κεφάλαιο αναλύεται η οργάνωση της συσκευής με χρήση του Web Manager.
Για να εισέλθει ο χρήστης στον Web Manager υπάρχουν δύο τρόποι. Ο ένας είναι μέσω ενός φυλλομετρητή (web browser), πληκτρολογώντας απλά την IP που έχει λάβει η συσκευή προηγουμένως. Ο δεύτερος είναι μέσα από τον ίδιο τον Device Installer, μέσω της διαδικασίας που ακολουθεί:

- Στο περιβάλλον του Device Installer, ο χρήστης αναζητά τον κατάλογο με τις διαθέσιμες συσκευές Lantronix.
- Ο χρήστης ανοίγει, τον κατάλογο Xport.
- Ο χρήστης κάνει διερεύνηση στα στοιχεία του συγκεκριμένου καταλόγου επιλέγοντας το + δίπλα από το εικονίδιο του Xport.
- Ο χρήστης επιλέγει τη συσκευή Xport που τον ενδιαφέρει, βάσει της MAC διεύθυνσής της.
- Στο παράθυρο που βρίσκεται στα δεξιά, ο χρήστης επιλέγει την καρτέλα Web Configuration.
- Για την προβολή του Web Manager μέσα από το περιβάλλον του Device Installer, ο χρήστης επιλέγει το κουμπί Go, ειδάλλως με την επιλογή External Browser, ο χρήστης έχει πρόσβαση στο περιβάλλον του Web Manager μέσω φυλλομετρητή.

Αμέσως μετά εμφανίζεται παράθυρο που ζητάει εισαγωγή ονόματος χρήστη και κωδικού πρόσβασης.

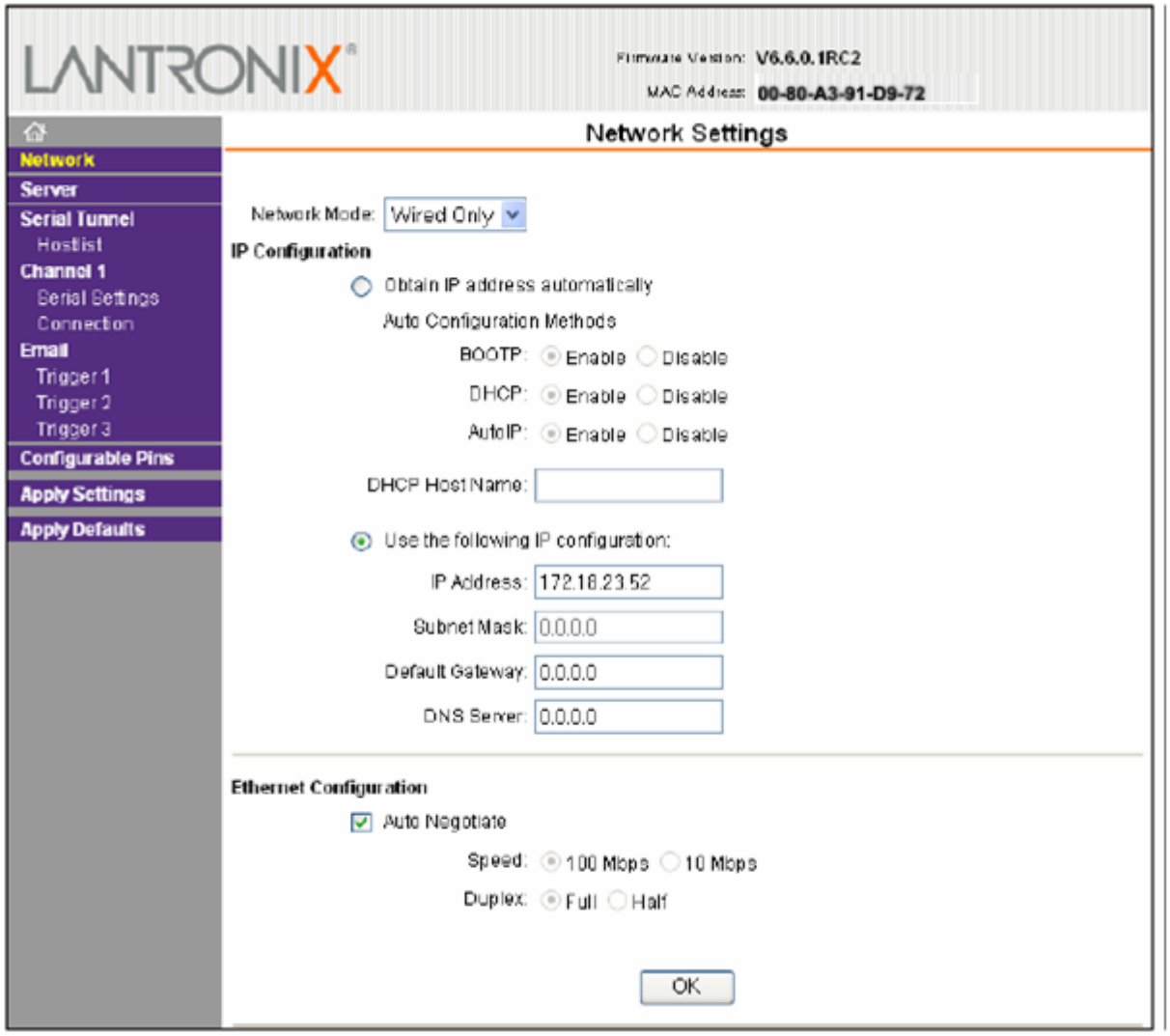
Σχήμα 3.3.1: Παράθυρο σύνδεσης χρήστη

Αν δεν έχει γίνει ήδη κάποια ρύθμιση για αυτά τα στοιχεία μέσω σειριακής ή telnet, τότε ο χρήστης αφήνει κενά αυτά τα πεδία και συνεχίζει πιέζοντας OK.
Τότε εμφανίζεται η αρχική σελίδα του Web Manager



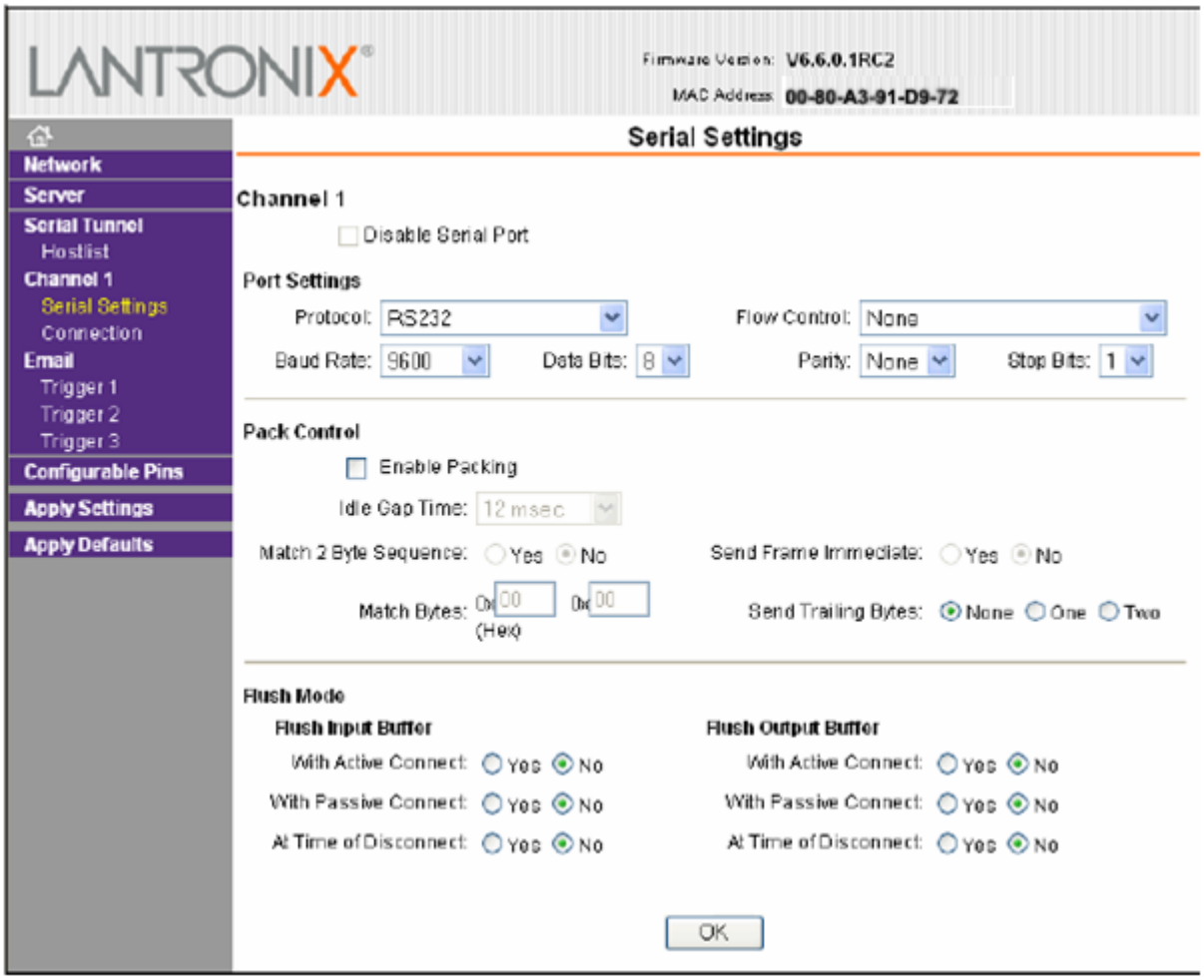
Σχήμα 3.3.2: Αρχική οθόνη παραμετροποίησης XPORT-05R

Για τις ανάγκες της τρέχουσας εργασίας, θα αναλυθούν μόνο τα πεδία που μεταβλήθηκαν στον Web Manager και όχι διεξοδικά, όλο το περιβάλλον αυτού.
Από τον κατάλογο Network, επιλέγει το Use the following IP configuration και ορίζει την 192.168.1.6 ως διεύθυνση στην οποία θα ανταποκρίνεται η συσκευή.



Σχήμα 3.3.3:Ρύθμιση Σύνδεσης IP

Από τον κατάλογο Channel 1 →Serial Settings, επιλέγει Baud Rate 9600 Kbps, Data bits 8, Flow Control None, Parity none, Stop bits 1.



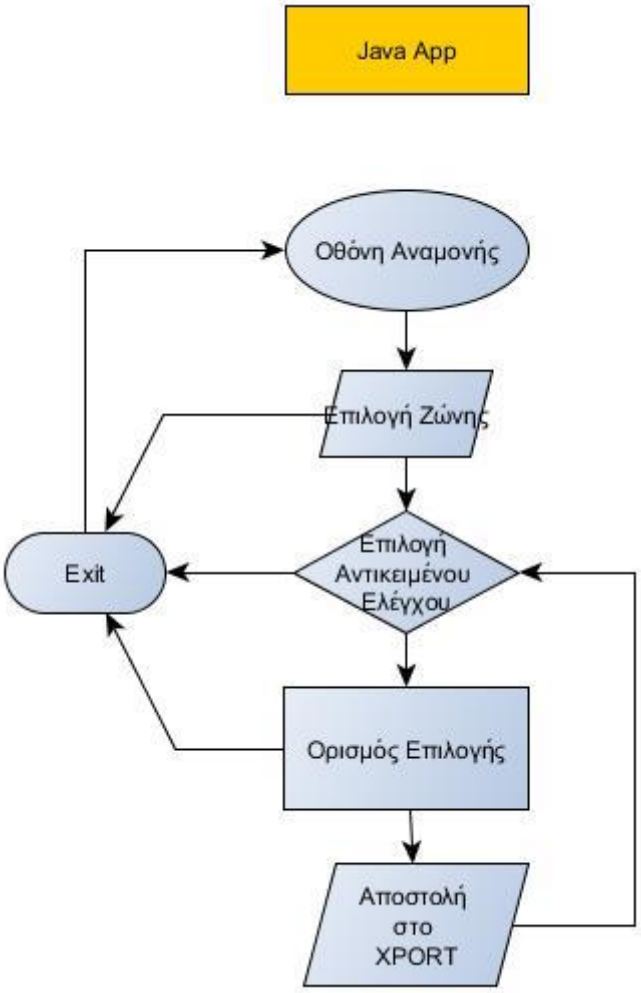
Σχήμα 3.3.4: Ρυθμίσεις σειριακής θύρας

3.4 ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΔΙΑΣΥΝΔΕΣΗΣ ΤΗΣ JAVA

Το περιβάλλον JAVA το οποίο έχει υλοποιηθεί είναι απλό και βασίζεται σε μία απλή διεπαφή χρήστη η οποία επιτρέπει τον έλεγχο της κατασκευής με το πάτημα των ανάλογων κουμπιών. Για να γίνει αυτό έχουν δημιουργηθεί οι απαραίτητες αντιστοιχίες έτσι ώστε όταν ο χρήστης πατάει τα ανάλογα κουμπιά στο παράθυρο που έχει δημιουργηθεί από την JAVA και βρίσκεται στην διεύθυνση IP την οποία χρησιμοποιεί το XPORT να στέλνει την ανάλογη εντολή ανοίγματος η κλεισίματος του εκάστωτε οργάνου ελέγχου που έχει συσχετιστεί με εκείνη την εντολή. Αυτό γίνεται εφόσον η ανάλογη εντολή «0» ή «1» ακολουθούμενη από την κατάσταση στην οποία είναι και τα υπόλοιπα κουμπιά μπουν με συγκεκριμένη σειρά σε μια οκτάδα δυαδικών ψηφίων. Αυτά με την σειρά τους στέλνονται

σειριακά στον επεξεργαστή προκειμένου να επεξεργαστεί τις απαιτούμενες καταστάσεις που ζητάει ο χρήστης.

Το διάγραμμα ροής που προκύπτει από τον προγραμματισμό της διεπαφής φαίνεται παρακάτω απλουστευμένο ως προς τα επίπεδα στα οποία υλοποιήθηκε η όλη σχεδίαση του προγράμματος.



ΚΕΦΑΛΑΙΟ 4

ΕΦΑΡΜΟΓΗ / ΥΛΟΠΟΙΗΣΗ

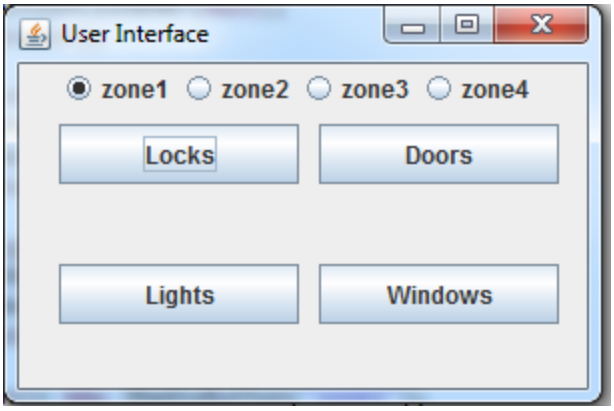
4.1 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ XPORT-05R

Σε αυτό το κομμάτι θα περιγραφεί η διαδικασία προγραμματισμού της διεπαφής χρήστη που δημιουργήθηκε καθώς και τον κώδικα που αυτή προϋποθέτει για την ολοκλήρωση της.

Στην οθόνη διεπαφής χρήστη όπως φαίνεται και στην παρακάτω εικόνα υπάρχουν τέσσερεις επιλογές ζώνης καθώς και τέσσερα κουμπιά λειτουργίας Toggle On/Off τα οποία ελέγχουν τα στοιχεία που είναι σχεδιασμένα να συνδεθούν και να ελέγχονται από το σύστημα.

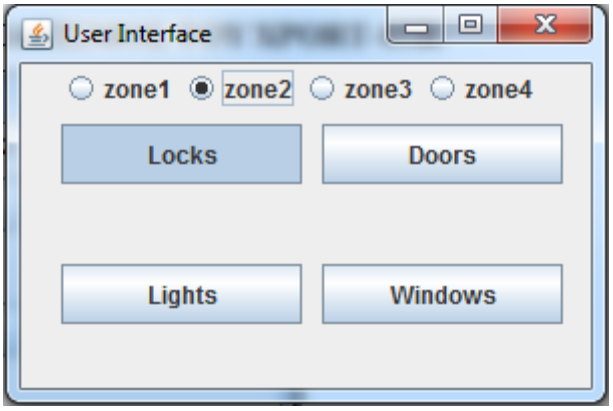
Η επιλογή διαφορετικής ζώνης έχει σαν σκοπό να μπορεί ο χρήστης να επιλέξει ανάμεσα στις τρείς πόρτες που διαθέτει ο επεξεργαστής και να εναλλάσσει τις εντολές ανάμεσα σε αυτές. Επιλέγοντας οποιαδήποτε από τις πρώτες τρείς πόρτες και μετά πατώντας το εκάστοτε κουμπί ο χρήστης δίνει την ανάλογη εντολή στο κάθε σημείο ελέγχου.

Όσον αφορά την τέταρτη ζώνη έχει σαν σκοπό να υλοποιήσει ακριβώς τις ίδιες εντολές με την διαφορά ότι αυτή δεν στέλνει σε μια συγκεκριμένη πόρτα του μικροελεγκτή μας αλλά στους δύο ακροδέκτες που υλοποιούν το πρωτόκολλο I^2C.



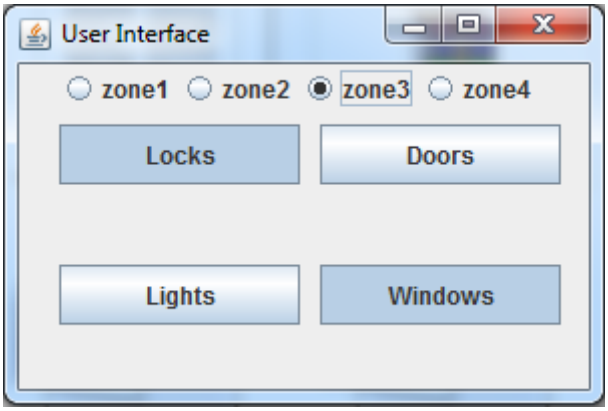
Σχήμα 4.1.1: Τελικό παράθυρο ελέγχου χρήστη

Όπως φαίνεται και παρακάτω εφόσον ο χρήστης το επιθυμεί μπορεί να αλλάξει την ζώνη ελέγχου και να δίνει διαφορετικές εντολές αναλόγως με τις ανάγκες που προκύπτουν.



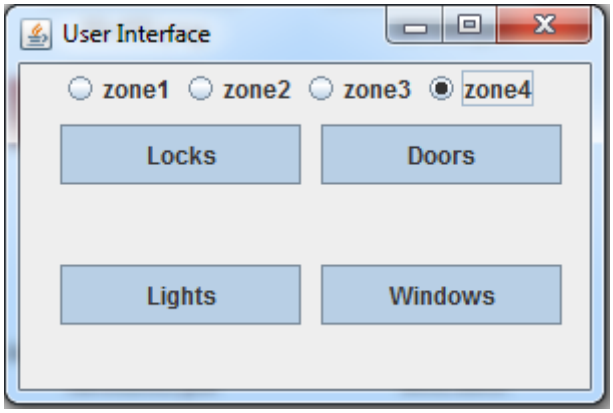
Σχήμα 4.1.2: Τελικό παράθυρο ελέγχου χρήστη με επιλογή εναλλακτικής ζώνης και έλεγχο μονού στοιχείου

Δίνεται επίσης η δυνατότητα να ελέγχεται παραπάνω από ένα κουμπί ταυτόχρονα σε κάθε ένα από τα σημεία ελέγχου.



Σχήμα 4.1.3: Τελικό παράθυρο ελέγχου χρήστη με επιλογή εναλλακτικής ζώνης και έλεγχο διπλού στοιχείου

Ο διαχωρισμός ανάμεσα στο «Ανοιχτό» και «Κλειστό» σύστημα ελέγχου έχει να κάνει με το κατά πόσο είναι πατημένο το κουμπί που διαχειρίζεται τις διάφορες εξωτερικές συσκευές.



Σχήμα 4.1.2: Τελικό παράθυρο ελέγχου χρήστη με επιλογή εναλλακτικής ζώνης και έλεγχο όλων των στοιχείων

Σε επίπεδο κώδικα παρατηρείται ότι είναι αρκετά απλός ο σχεδιασμός της συγκεκριμένης διεπαφής ελέγχου και είναι εύκολα διαχειρίσιμο και αντιληπτό από τον εκάστωτε χρήστη. Αυτό δίνει την δυνατότητα στο σύστημα να μπορεί να επεκταθεί περεταίρω για μελλοντικές κατασκευές μεγαλύτερης πολυπλοκότητας αλλά και χρηστικής λειτουργίας.

Στην αρχή του κώδικα όπως και στις περισσότερες γλώσσες προγραμματισμού που διδάσκονται στο ΤΕΙ το πρόγραμμα αρχίζει με τις αρχικοποιήσεις και τις δηλώσεις των βιβλιοθηκών οι οποίες θα χρησιμοποιηθούν προκειμένου να εκτελεστεί η συγκεκριμένη εφαρμογή.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
```

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JToggleButton;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
```

Στην συνέχεια αρχικοποιούνται οι θέσεις τις οποίες θα πρέπει να έχει αρχικά η διεπαφή με προεπιλογές την ζώνη 1, και όλα τα συστήματα απενεργοποιημένα.

```
public class panel_instance() {
   zone=1;
   windows=false;
   doors=false;
   locks=false;
   lights=false;
   code={A,A,0}
}
```

Με αυτή την συνάρτηση ρυθμίζεται ο κωδικός που θα αποστέλλεται με την εντολή setCode μέσω της σειριακής. Τα στοιχεία που θα αποσταλούν θα είναι της μορφής χαρακτήρων για τις ζώνες και τα κουμπιά με τιμές από A έως D και 0 ή 1 για το εάν είναι ενεργοποιημένο η όχι το κάθε κουμπί. Επομένως εάν χρειάζεται για παράδειγμα η ρύθμιση των κλειδαριών της τρίτης ζώνης να είναι «Κλειστά» τότε θα αποστείλει τον κωδικό «CC1»

```
public void setZone(int newZone) {
   zone = newZone;
   setCode(zone,code[1],code[2])
}
```

Οι επόμενες τέσσερεις συναρτήσεις στέλνουν τον εκάστωτε χαρακτήρα αναλόγως με το εάν το κουμπί ελέγχου της συσκευής είναι πατημένο η όχι.

```
public void setLights(boolean newLight) {
   lights = newLight;
   if (lights==true)
      {setCode(code[0],"A","1")}
      else setCode(code[0],"A","0");
}

public void setDoors(boolean newDoor) {
   doors = newDoor;
if (doors==true)
      {setCode(code[0],"B","1")}
      else setCode(code[0],"B","0");
}
```

```
public void setLock(boolean newLock) {
   locks = newLock;
   if (locks==true)
      {setCode(code[0],"C","1")}
      else setCode(code[0],"C","0");
}

public void setWindow(boolean newWin) {
   windows = newWin;
   if (windows==true)
      {setCode(code[0],"D","1")}
      else setCode(code[0],"D","0");
}
```

Η παρακάτω συνάρτηση ρυθμίζει τα στοιχεία που θα δέχεται η συνάρτηση setCode τα οποία τα διαβάζει σε μορφή χαρακτήρων όπως φαίνεται και παρακάτω.

```
public void setCode(char a , char b ,char c) {
   code[0]=a;
   code[1]=b;
   code[2]=c;
}
}
```

Στην συνέχεια του προγράμματος σχεδιάζεται το J(ava)Frame το οποίο θα αποτελέσει την διεπαφή που θα χρησιμοποιηθεί στο σύστημα. Για να γίνει αυτό δημιουργούνται τα τέσσερα κουμπιά με τις ανάλογες ονομασίες τους, τους δίνεται η θέση την οποία θα έχουν στο παράθυρο καθώς και το μέγεθος τους και στην συνέχεια δημιουργείται η λειτουργία που θα εκτελέσει το κουμπί όταν λάβει μία Action δηλαδή όταν αλλάξει η κατάσταση του από ανοιχτό σε κλειστό ή αντίστροφα. Για παράδειγμα όταν το κουμπί για τα παράθυρα είναι μη πατημένο (δηλαδή σε κατάσταση «false») να στέλνει στην μεταβλητή window την τιμή 0 αλλιώς να στέλνει την τιμή 1. Με την ίδια σειρά και λογική υλοποιούνται και τα υπόλοιπα στοιχεία της σχεδίασης των κουμπιών ελέγχου.

```
   public class example extends JFrame
   {
      public example() { initUI();

                  }


      public final void initUI()
         {
            JPanel panel = new JPanel();
            getContentPane().add(panel);
            panel.setLayout(null);
            panel_instance panel1 = new panel_instance();
```

```
JToggleButton windowsButton = new JToggleButton("Windows");
windowsButton.setBounds(150, 100, 120, 30);
windowsButton.
windowsbutton.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e)
   {
      //Execute when button is pressed
      if(panel1.windows=="false")
      {panel1.setWindow(true);}
      else panel1.setWindow(false);
   }
});
```

```
JToggleButton doorsButton = new JToggleButton("Doors");
doorsButton.setBounds(150, 30, 120, 30);
doors
button.addActionListener(new ActionListener() {

   public void actionPerformed(ActionEvent e)
   {
      //Execute when button is pressed
      if(panel1.doors=="false")
      {panel1.setdoors(true);}
      else panel1.setdoors(false);
   }
});
panel.add(doorsButton);
```

```
JToggleButton locksButton = new JToggleButton("Locks");
locksButton.setBounds(20, 30, 120, 30);
locksbutton.addActionListener(new ActionListener() {

   public void actionPerformed(ActionEvent e)
   {
      //Execute when button is pressed
      if(panel1.locks=="false")
      {panel1.setlocks(true);}
      else panel1.setlocks(false);
   }
});
panel.add(locksButton);
```

```
JToggleButton lightsButton = new JToggleButton("Lights");
lightsButton.setBounds(20, 100, 120, 30);
lights
button.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e)
   {
      //Execute when button is pressed
      if(panel1.lights=="false")
      {panel1.setlights(true);}
      else panel1.setlights(false);
   }
});
panel.add(lightsButton);
```

Στο σημείο αυτό ρυθμίζονται τα στοιχεία που επιλέγουν ανάμεσα στις διαφορετικές ζώνες. Όπως μπορεί κανείς να παρατηρήσει εδώ δεν γίνεται κάποια συγκεκριμένη διαδικασία μετά το πάτημα της κάθε ζώνης δεδομένου ότι τα δεδομένα στέλνονται μονάχα με το πάτημα τον κουμπιών ελέγχου.

Αντίστοιχα με παραπάνω τίθεται η ονομασία και ο τύπος του κουμπιού, το μέγεθος και η θέση του στο παράθυρο και την διεργασία την οποία επιθυμείται να κάνει και στην περίπτωσή αυτή είναι καμία δηλαδή «null».

```
JRadioButton zone1= new JRadioButton("zone1");
zone1.setBounds(20, 2, 60, 20);
zone1.addActionListener(null);
panel.add(zone1);

JRadioButton zone2= new JRadioButton("zone2");
zone2.setBounds(80, 2, 60, 20);
zone2.addActionListener(null);
panel.add(zone2);

JRadioButton zone3= new JRadioButton("zone3");
zone3.setBounds(140, 2, 60, 20);
zone3.addActionListener(null);
panel.add(zone3);

JRadioButton zone4= new JRadioButton("zone4");
zone4.setBounds(200, 2, 60, 20);
zone4.addActionListener(null);
panel.add(zone4);
```

Εδώ συσχετίζονται τα κουμπιά εναλλαγής ζώνης σε μία ενιαία ομάδα προκειμένου να μην μπορεί να είναι πατημένο παραπάνω από ένα την φορά και να προκαλέσει σύγχυση στο σύστημα και κατ’επέκταση σφάλμα στην αποστολή και εκτέλεση των εντολών.

```
ButtonGroup zone = new ButtonGroup( );
zone.add(zone1);
zone.add(zone2);
zone.add(zone3);
zone.add(zone4);
```

Τελειώνοντας το πρόγραμμα τίθεται ο τίτλος και οι διαστάσεις του παραθύρου ελέγχου, την διαδικασία που θα εκτελέσει στο πάτημα του κουμπιού κλεισίματος στο πάνω δεξιά μέρος της οθόνης διεπαφής και στην συνέχεια δηλώνεται ότι η όλη διεργασία που υλοποιείται παραπάνω θα γίνεται σε ένα παράθυρο το οποίο θα είναι εμφανές θέτοντας την παράμετρο ex.setVisible αληθή (true).

```
            setTitle("User Interface");
            setSize(300, 200);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
         }
      public static void main(String[] args)
      {
         example ex = new example(); ex.setVisible(true);
         }
}
```

4.2 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΕΠΕΞΕΡΓΑΣΤΗ

Όπως και παραπάνω ο προγραμματισμός του επεξεργαστή έγινε με το πρόγραμμα Keil μVision της Maxim.Στην συνέχεια θα εξηγηθεί η δομή που έχει το πρόγραμμα καθώς και ο κώδικας που αναπτύχθηκε για την λειτουργία της κατασκευής. Στο πρώτο κομμάτι τοποθετούντε οι βιβλιοθήκες που χρειάζονται να συμπεριληφθούν στον κώδικα όπως την βιβλιοθήκη του επεξεργαστή, την βιβλιοθήκη για το I2C πρωτόκολλο και την βιβλιοθήκη που περιέχει τις ρυθμήσεις για την οθόνη LCD.

CPU CODE

```
#include <DS89C4XX.h>    //DS89C450 Microcontroller
#include <intrins.h>
#include <stdio.h>
#define  _Nop()  _nop_()
#include <string.h>
#include <i2c.h>
```

```
#include <lcd.h>
```

Στην συνέχεια υπάρχουν οι διάφορες υπορουτίνες που θα χρησιμοποιηθούν σε ξεχωριστά κομμάτια προκειμένου να μπορεί το πρόγραμμα να ανατρέξει σε αυτές κατά την διάρκεια επίκλησης τους από αυτό. Οι υπορουτίνες είναι οι εξής:
Ρουτίνα χρονοκαθυστέρησης η οποία παίρνει τιμές σε μSec και κάνει το πρόγραμμα να περιμένει για τον εκάστοτε χρόνο.

Delay

```
void delay(int count)
{
   int i,j;

   for (i=0;i<count;i++){
   for (j=0;j<1200;j++); }
}
```

Αρχικοποίηση της οθόνης και το πρώτο μήνυμα που θα εμφανίζεται σε αυτήν.

LCD Initialisation

```
void norm_show(void)
{
 lcd_clear();

 printf("\n\r\n\rSYSTEM OPERATING\n\r");
}
```

Στο συγκεκριμένο σημείο του κώδικα αντιστοιχούνται τα ονόματα που θα δοθούν στους ακροδέκτες του κώδικα τα οποία θα χρησιμοποιηθούν μετέπειτα στον κώδικα για την λειτουργία του.
Οι ακροδέκτες που απευθύνονται στην εκτέλεση του πρωτοκόλλου I2C με την γραμμή Data και Clock αντίστοιχα.

```
sbit SDA=P0^1;
sbit SCL=P0^0;
```

Στην συνέχεια η ρύθμιση για τις εξόδους της σχεδίασης για την πόρτα 1,2 και 3. Η πόρτα 4 αντιστοιχεί στην έξοδο που μεταβιβάζει τα δεδομένα του μέσω του πρωτοκόλλου I2C.

Port #1 Definition Names

```
sbit LT1=P2^7;
sbit LT1FDB=P2^6;
sbit DR1=P2^5;
sbit DR1FDB=P2^4;
sbit LK1=P2^3;
sbit LK1FDB=P2^2;
sbit WD1=P2^1;
sbit WD1FDB=P2^0;
```

Port #2 Definition Names

```
sbit LT2=P3^0;
sbit LT2FDB=P3^1;
sbit DR2=P3^2;
sbit DR2FDB=P3^3;
sbit LK2=P3^4;
sbit LK2FDB=P3^5;
sbit WD2=P3^6;
sbit WD2FDB=P3^7;
```

Port #3 Definition Names

```
sbit LT3=P1^0;
sbit LT3FDB=P1^1;
sbit DR3=P1^4;
sbit DR3FDB=P1^5;
sbit LK3=P1^6;
sbit LK3FDB=P1^7;
```

Έχοντας παραμετροποιήσει τους ακροδέκτες του επεξεργαστή συνεχίζουμε με την ρύθμιση των ρουτινών που θα διαχειρίζονται το πρωτόκολλο I2C.

I2C buffer

```
void I2CBitDly()
{
unsigned int time_end = 10;
unsigned int index;
for (index = 0; index < time_end; index++);
return;
}
```

I2C START

```
void I2CSendStart()
{
  SDA = 1;            // i2c start bit sequence
  I2CBitDly();
  SCL = 1;
  I2CBitDly();
  SDA = 0;
  I2CBitDly();
  SCL = 0;
  I2CBitDly();
}
```

I2C Stop

```
void I2CSendStop()
{
  SDA = 0;            // i2c stop bit sequence
  I2CBitDly();
  SCL = 1;
  I2CBitDly();
  SDA = 1;
```

```
  I2CBitDly();
}
```

Set SCL high, and wait for it to go high

```
void I2CSCLHigh(void)
{
register int err;
SCL = 1;
while (! SCL)
{
err++;
if (!err)
{
return;
}
}
}
```

Send function

```
void I2CSendByte(unsigned char bt)
{
register unsigned char i;
for (i=0; i<8; i++)
{
if (bt & 0x80) SDA = 1; // Send each bit, MSB first changed 0x80 to 0x01
else SDA = 0;
I2CSCLHigh();
I2CBitDly();
SCL = 0;
I2CBitDly();
bt = bt << 1;
}
SDA = 1; // Check for ACK
```

```
I2CBitDly();
I2CSCLHigh();
I2CBitDly();
if (SDA)
SCL = 0;
I2CBitDly();
SDA = 1; // end transmission
SCL = 1;
}
```

Transmit to SDA

```
void I2CSendAddr(unsigned char addr, unsigned char rd)
{
I2CSendStart();
I2CSendByte(addr+rd); // send address byte
}
```

Receive from SDA

```
unsigned char I2CGetByte(unsigned char lastone) // last one == 1 for last byte; 0 for any other byte
{ register unsigned char i, res;
res = 0;
for (i=0;i<8;i++) // Each bit at a time, MSB first
{ I2CSCLHigh();
I2CBitDly();
res *= 2;
if (SDA) res++;
SCL = 0;
I2CBitDly();
} SDA = lastone; // Send ACK according to 'lastone'
I2CSCLHigh();
I2CBitDly();
SCL = 0;
```

```
SDA = 1; // end transmission
SCL=1;
I2CBitDly();
return(res);
}
```

SERIAL PORT

```
void Tx(unsigned char c)
{
   while (TI_0==0);
   TI_0=0;
   SBUF0=c;
}
```

Digital Transfer

```
void Tx_char(unsigned char c)
{
   unsigned char s[4];
   int i;

   c=c&0xff;
   sprintf(s,"%#x",(int)c);        // write into string.
   for (i=0;i<4;i++)
    Tx(s);
}
```

String port

```
void Uart_Tx2(unsigned char x)
{
   while (TI_0==0);
   TI_0=0;
```

```
   SBUF0=x;
}
```

Send String

```
void sendstring (char *String)
{
   int i=0;    //set counter to 0
   while(String)
   {
      Uart_Tx2(String[i++]);
   }
}
```

Σε αυτό το κομμάτι είναι τον βασικός κώδικας της κατασκευής.

Όπως επεξηγήθηκε και παραπάνω το σύστημα περιμένει μέσω της σειριακής του να λάβει τον εκάστωτε χαρακτήρα από το XPORT ο οποίος αναλόγως με το ποιος θα είναι θα εκτελεί και την ανάλογη εντολή. Ο έλεγχος γίνεται με διάφορες ρουτίνες οι οποίες ελέγχουν τον χαρακτήρα που λαμβάνει το σύστημα και εκτελούν τις εντολές που χρειάζεται μέχρι να φτάσουν στο σημείο να δώσουν εντολή στις εξόδους του συστήματος και να ενημερώσουν τον χρήστη για το αποτέλεσμα με την βοήθεια της LCD οθόνης.

Main Code

```
void main ()
{
    unsigned char c;
    unsigned char addr;
    unsigned char lastone;
    unsigned char rd;
    SCON1 = 0x52;
    SCON0 = 0x52;
    TMOD  = 0x20;
    TCON  = 0x69;
    TH1   = 0xFD;
```

```
delay(500);
norm_show();
printf("WELCOME!SYSTEM STARTUP...\n\r");
delay(2000);
lcd_clear();
```

Εδώ μπαίνει το σύστημα σε έναν ατέρμονο βρόγχο και περιμένει τον χαρακτήρα προκειμένου να μεταβεί ανάμεσα στις διάφορες ζώνες και στην συνέχεια στο κάθε σημείο ελέγχου.

```
while(1)
{
norm_show();
   c=getchar();
    if (c=='A')           //epilogi zwnis 1
   {
    c=getchar();
    if (c=='A')          //epilogi lights1
    {
     c=getchar();
     if (c=='0') LT1=0;
     else LT1=1;

     delay(500);
```

Εδώ χρησιμοποιήθηκε ανάστροφη λογική στους ακροδέκτες του Feedback δεδομένου ότι είναι ευκολότερο να γειωθούν οι επαφές των αισθητήρων σε μεγάλη απόσταση παρά να παρέχεται η απαραίτητη τάση και έτσι να ελεγχθεί κατά πόσο ανταποκρίθηκε το σύστημα.

```
     if (LT1FDB==1)
     {
      lcd_clear();
      delay(1000);
      printf("\n\rERROR: LIGHTS1 NOT RESPONDING\n\r");
     }
     else
```

```
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLIGHTS1 OK!\n\r");
 }
}
```

Η ίδια λογική ακολουθείται για τους υπόλοιπους αισθητήρες αλλά και για τις υπόλοιπες ζώνες.

```
if (c=='B')          //epilogi door1
{         c=getchar();
 if (c=='0') DR1=0;
 else DR1=1;
 delay(500);
 if (DR1FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR1 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR1 OK!\n\r");
 }
}
if (c=='C')          //epilogi lock1
{
 c=getchar();

 if (c=='0') LK1=0;
 else LK1=1;
  delay(500);
```

```
if (LK1FDB==1)
{
 lcd_clear();
 delay(1000);
 printf("\n\rERROR: LOCK1 NOT RESPONDING\n\r");
}
else
{
 lcd_clear();
 delay(1000);
 printf("\n\rLOCK1 OK!\n\r");
}
}
if (c=='D')          //epilogi window1
{
c=getchar();

if (c=='0') WD1=0;
else WD1=1;
delay(500);
```

```
if (WD1FDB==1)
{
 lcd_clear();
 delay(1000);
 printf("\n\rERROR: WINDOW1 NOT RESPONDING\n\r");
}
else
{
 lcd_clear();
 delay(1000);
 printf("\n\rWINDOW1 OK!\n\r");
```

```
      }
     }
    }
```

Επιλογή ζώνης 2:

```
    else if(c=='B')         //epilogi zwnis2
    {
     c=getchar();

     if (c=='A')          //epilogi lights2
     {
      c=getchar();

      if (c=='0') LT2=0;
      else LT2=1;
      delay(500);
```

```
      if (LT2FDB==1)
      {
       lcd_clear();
       delay(1000);
       printf("\n\rERROR: LIGHTS2 NOT RESPONDING\n\r");
      }
      else
      {
       lcd_clear();
       delay(1000);
       printf("\n\rLIGHTS2 OK!\n\r");
      }
     }
     if (c=='B')          //epilogi door2
```

```
{
 c=getchar();

 if (c=='0') DR2=0;
 else DR2=1;

 delay(500);
 if (DR2FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR2 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR2 OK!\n\r");
 }
}
if (c=='C')          //epilogi lock2
{
 c=getchar();

 if (c=='0') LK2=0;
 else LK2=1;

 delay(500);
 if (LK2FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: LOCK2 NOT RESPONDING\n\r");
```

```
        }
        else
        {
         lcd_clear();
         delay(1000);
         printf("\n\rLOCK2 OK!\n\r");
        }
       }
       if (c=='D')           //epilogi window2
       {
        c=getchar();
        if (c=='0') WD2=0;
        else WD2=1;

        delay(500);
        if (WD2FDB==1)
        {
         lcd_clear();
         delay(1000);
         printf("\n\rERROR: WINDOW2 NOT RESPONDING\n\r");
        }
        else
        {
         lcd_clear();
         delay(1000);
         printf("\n\rWINDOW2 OK!\n\r");
        }
       }
      }
```

Επιλογή ζώνης 3:

```
      else if(c=='C')         //epilogi zwnis3
      {
       c=getchar();
```

```
if (c=='A')          //epilogi lights3
{
 c=getchar();
 if (c=='0') LT3=0;
 else LT3=1;
 delay(500);
 if (LT3FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: LIGHTS3 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLIGHTS3 OK!\n\r");
 }
}
if (c=='B')          //epilogi door3
{
 c=getchar();

 if (c=='0') DR3=0;
 else DR3=1;

 delay(500);
 if (DR3FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR3 NOT RESPONDING\n\r");
 }
```

```
        else
        {
         lcd_clear();
         delay(1000);
         printf("\n\rDOOR3 OK!\n\r");
        }
       }
      if (c=='C')           //epilogi lock3
      {
       c=getchar();

       if (c=='0') LK3=0;
       else LK3=1;

       delay(500);
       if (LK3FDB==1)
       {
        lcd_clear();
        delay(1000);
        printf("\n\rERROR: LOCK3 NOT RESPONDING\n\r");
       }
       else
       {
        lcd_clear();
        delay(1000);
        printf("\n\rLOCK3 OK!\n\r");
       }
      }
     }
```

Εδώ φαίνεται η επιλογή της τέταρτης ζώνης η οποία είναι και αυτή που υλοποιεί το πρωτόκολλο I2C. Σε αυτή την ζώνη ακολουθείται μία διαδικασία η οποία διαβάζει την εκάστοτε εντολή αρχικοποιεί τον δίαυλο επικοινωνίας, στέλνει εντολή έναρξης σε όλες τις συσκευές, μετατρέπει τον χαρακτήρα σε ένα Byte

δεδομένων, διαβάζει την διεύθυνση του παραλήπτη και επιβεβαιώνει ότι αυτός είναι έτοιμος να λάβει, αποστέλλει την πληροφορία και περιμένει επιβεβαίωση και στο τέλος ενημερώνει το σύστημα και κλείνει τον δίαυλο επικοινωνίας μεταξύ των συσκευών. Λόγω του ότι είναι σύστημα με έναν μόνο Master δεν μπορεί κάποια από τις άλλες συσκευές να ζητήσει την επανέναρξη της συνεδρίας.
Επίσης αντίστοιχος κώδικας υπάρχει σε κάθε παραλήπτη προκειμένου να είναι σε θέση να συγχρονιστεί με τον αποστολέα και να γίνει επιτυχώς η μετάδοση των δεδομένων. Με βάση τα παραπάνω ο κώδικας θα γίνει ως εξής:

```
if (c=='D')            //epilogi zwnis 4 (I2C)
  {
   c=getchar();
   if (c=='A')          //epilogi lights4
   {
    c=getchar();

    if (c=='0')
   {     I2CBitDly();
         I2CSCLHigh();
         I2CSendStart();
         Tx_char(0);
         I2CSendAddr(addr,rd);
         I2CGetByte(lastone);
         I2CSendStop();
         delay(500);          }
    else          {
         I2CBitDly();
         I2CSCLHigh();
         I2CSendStart();
         Tx_char(1);
         I2CSendAddr(addr,rd);
         I2CGetByte(lastone);
         I2CSendStop();
         delay(500);          };
```

```
        delay(500);
```

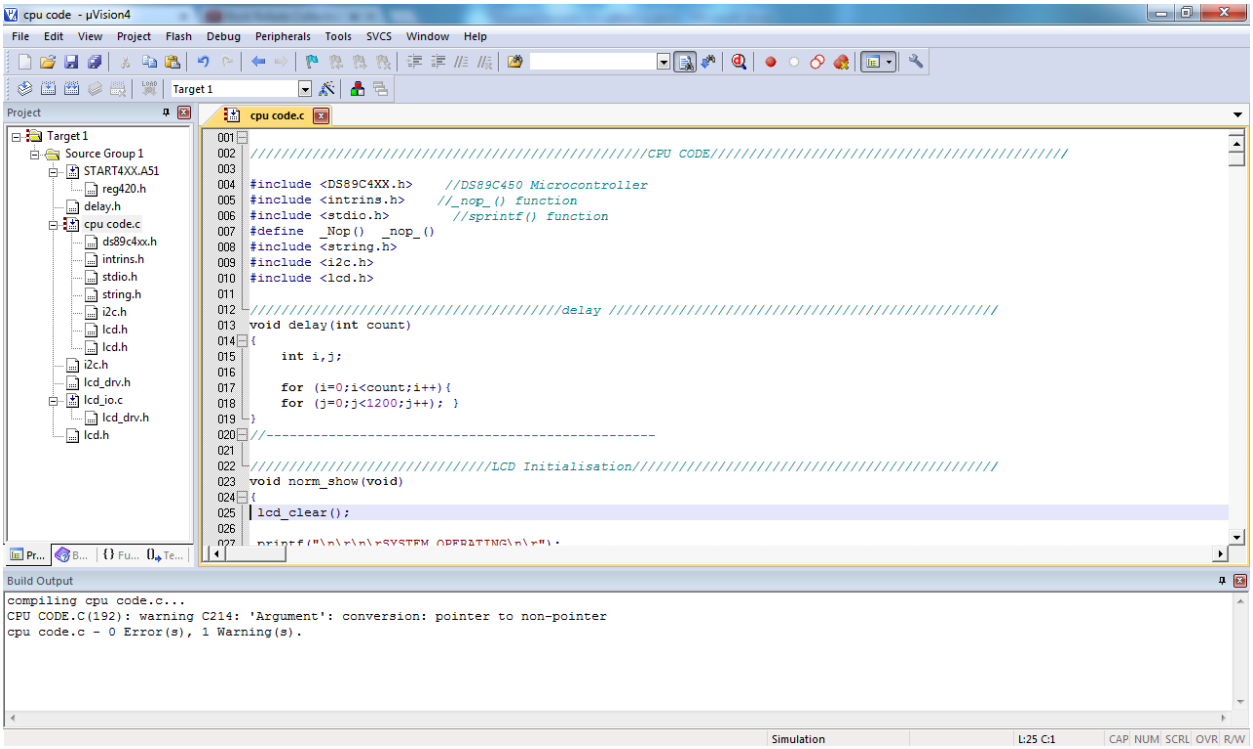
```
      if (c==1)
      {
       lcd_clear();
       delay(1000);
       printf("\n\rERROR: LIGHTS4 NOT RESPONDING\n\r");
      }
      else
      {
       lcd_clear();
       delay(1000);
       printf("\n\rLIGHTS4 OK!\n\r");
      }
     }
     if (c=='B')          //epilogi door4
     {          c=getchar();
       if (c=='0')
    {      I2CBitDly();
           I2CSCLHigh();
           I2CSendStart();
           Tx_char(0);
           I2CSendAddr(addr,rd);
           I2CGetByte(lastone);
           I2CSendStop();
           delay(500);
          }
      else
        {
           I2CBitDly();
           I2CSCLHigh();
           I2CSendStart();
           Tx_char(1);
```

```
        I2CSendAddr(addr,rd);
        I2CGetByte(lastone);
        I2CSendStop();
        delay(500);
        };
 delay(500);
 if (c==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR4 NOT RESPONDING\n\r");
 }
 else        {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR4 OK!\n\r");
 }
}
if (c=='0')
{
        I2CBitDly();
        I2CSCLHigh();
        I2CSendStart();
        Tx_char(0);
        I2CSendAddr(addr,rd);
        I2CGetByte(lastone);
        I2CSendStop();
        delay(500);
       }
 else
   {
        I2CBitDly();
        I2CSCLHigh();
```

```
        I2CSendStart();
        Tx_char(1);
        I2CSendAddr(addr,rd);
        I2CGetByte(lastone);
        I2CSendStop();
        delay(500);
        };
  delay(500);
          if (c==1)
  {
   lcd_clear();
   delay(1000);
   printf("\n\rERROR: LOCK4 NOT RESPONDING\n\r");
  }
  else
  {
   lcd_clear();
   delay(1000);
   printf("\n\rLOCK4 OK!\n\r");
  }
 }
 if (c=='D')          //epilogi window4
 {        c=getchar();
  if (c=='0')
 {      I2CBitDly();
        I2CSCLHigh();
        I2CSendStart();
        Tx_char(0);
        I2CSendAddr(addr,rd);
        I2CGetByte(lastone);
        I2CSendStop();
        delay(500);
       }
```

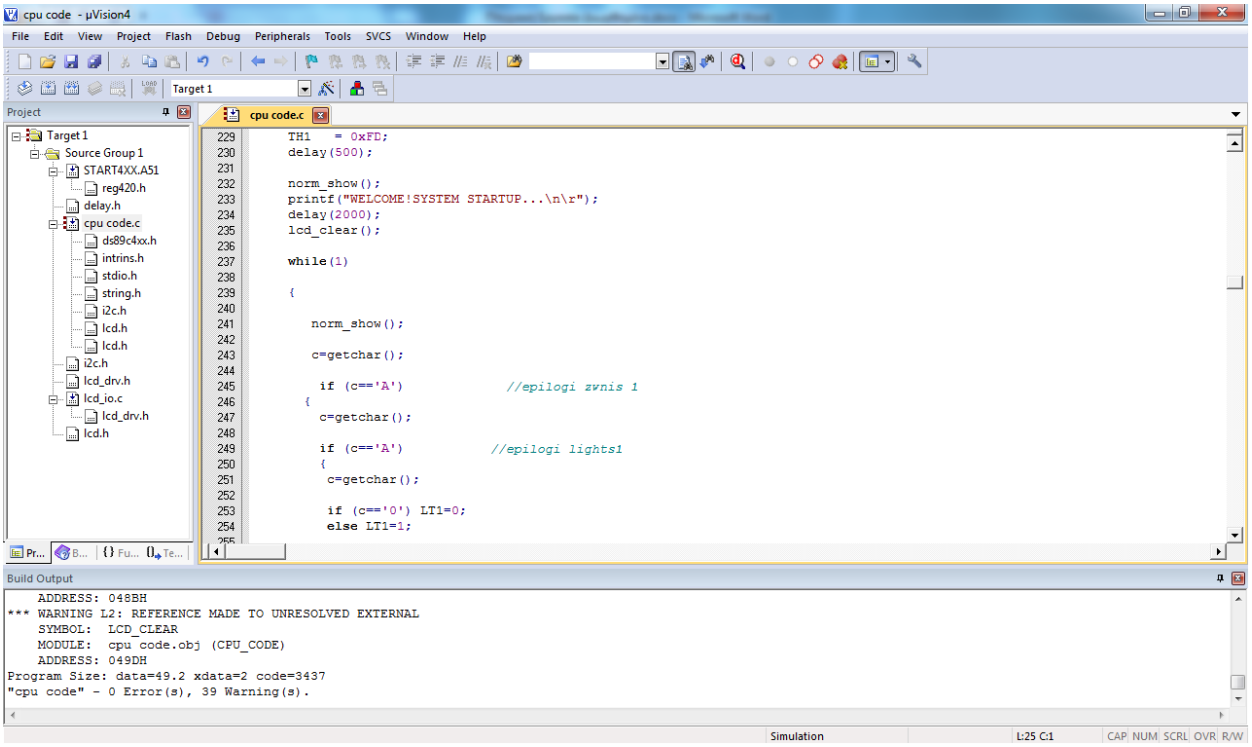
```
        else
          {
             I2CBitDly();
             I2CSCLHigh();
             I2CSendStart();
             Tx_char(1);
             I2CSendAddr(addr,rd);
             I2CGetByte(lastone);
             I2CSendStop();
             delay(500);
             };
        delay(500);
        if (c==1)
        {          lcd_clear();
         delay(1000);
         printf("\n\rERROR: WINDOW4 NOT RESPONDING\n\r");
        }         else
        {          lcd_clear();
         delay(1000);
         printf("\n\rWINDOW4 OK!\n\r");
        }
       }
     }
 }
```

Με τον παραπάνω κώδικα και την βοήθεια του μVision ελέγχθηκε για τυχόν λάθη στην συγγραφή του αλλά και για την δημιουργία του απαιτούμενου αρχείου για να το φορτωθεί στον μικροελεγκτή. Μετά τις διορθώσεις που χρειάστηκε να γίνουν μέχρι να ολοκληρωθεί ο κώδικας η μετάφραση του κώδικα από τον μεταγλωττιστή απέδωσε τα παρακάτω αποτελέσματα



Σχήμα 4.2.1: Αρχική οθόνη του Eclipse

Μετά την μεταγλώττιση ακολούθησε η δημιουργία του απαραίτητου αρχείου με την βοήθεια του κουμπιού Build το οποίο δημιούργησε εφόσον δεν υπήρχαν σφάλματα το απαραίτητο αρχείο για την ανάγνωση του από τον επεξεργαστή.



Σχήμα 4.2.2: Σύνταξη του κώδικα για την οθόνη ελέγχου χρήστη

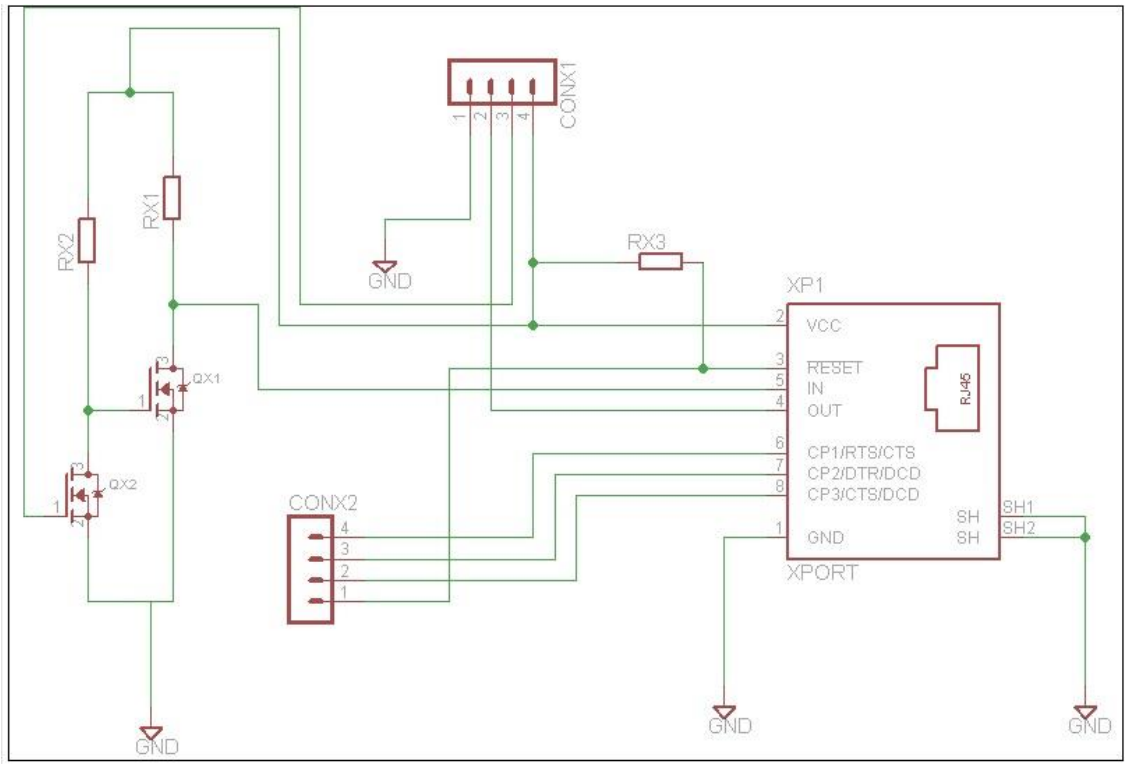
Έχοντας λοιπόν κανένα σφάλμα μπορεί ο χρήστης να προχωρήσει στην φόρτωση του κώδικα με τον τρόπο που επεξηγήθηκε στο κεφάλαιο 2

4.3 ΣΧΕΔΙΑΣΜΟΣ ΤΗΣ ΠΛΆΚΕΤΑΣ

Όπως φαίνεται και παραπάνω ο σχεδιασμός της πλακέτας πραγματοποιήθηκε με το πρόγραμμα EAGLE της CADSoft. Δημιουργήθηκαν αρχικά τρία συστήματα πριν αυτά ενσωματωθούν τελικά σε μία μεγαλύτερη και ενιαία σχεδίαση. Τα συστήματα αυτά αποτελούνται από:

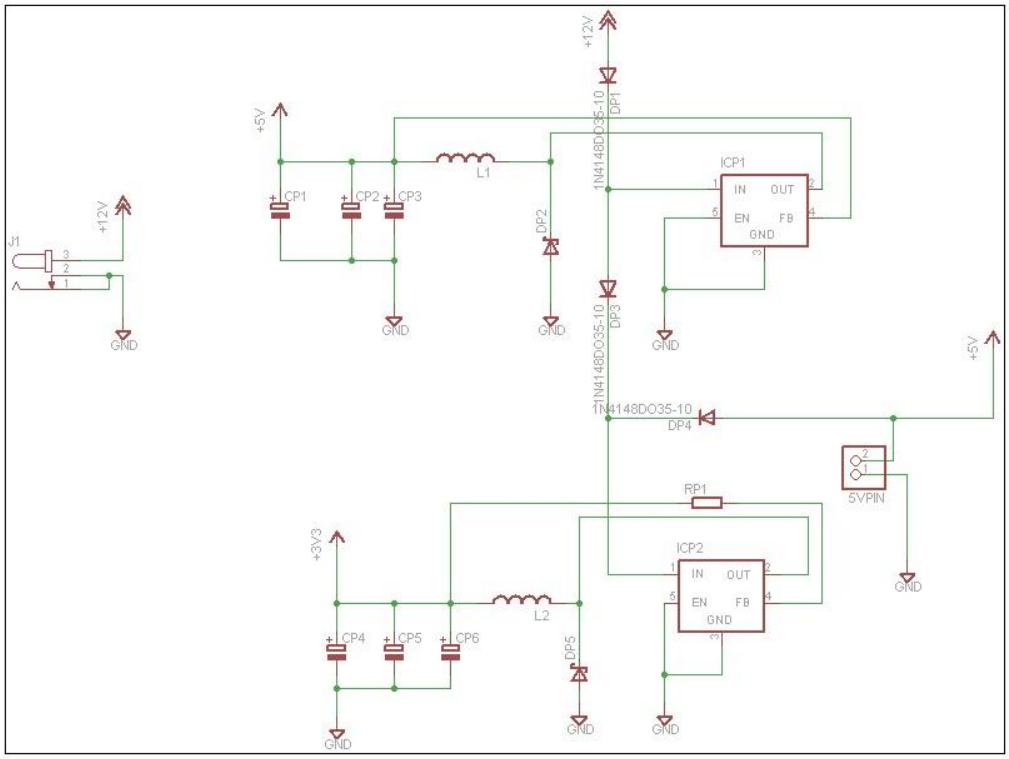
- Την πλακέτα τροφοδοσίας του XPORT-05R
- Την κεντρική τροφοδοσία του συστήματος μας
- Το κύκλωμα μετάβασης του επεξεργαστή σε κατάσταση προγραμματισμού
- Την πλακέτα του επεξεργαστή με τις εξόδους για τα περιφερικά του

Όσον αφορά την πλακέτα τροφοδοσίας εννοείται η πλακέτα που δέχεται 12Volt από μια εξωτερική πηγή, τα παρέχει στο σύστημα, και στην συνέχεια υποβιβάζει την συγκεκριμένη τάση σε 5 και 3.3volt. Αυτό συμβαίνει γιατί ο επεξεργαστής τροφοδοτείται στα 5Volt καθώς και τα σήματα που στέλνει είναι της ίδιας στάθμης. Ο λόγος που χρησιμοποιήθηκε και η τροφοδοσία 3.3Volt είναι για να τροφοδοτήσει το XPORT λόγο του ότι δεν επιδέχεται την στάθμη 5Volt όπως ο επεξεργαστής. Εκτός από τα σήματα τροφοδοσίας που λαμβάνει το XPORT και τα σήματα επικοινωνίας με τον επεξεργαστή είναι της στάθμης 5Volt γεγονός που τα καταστεί ακατάλληλα για το XPORT με κίνδυνο να το καταστρέψει. Για τον λόγο αυτό χρησιμοποιήθηκαν τα σήματα εισόδου προς το XPORT εφόσον περάσουν από δύο MOSFET Transistor προκειμένου να κατεβεί η τάση στο επιθυμητό και επιτρεπτό επίπεδο. Σε αντίθεση με το XPORT ο επεξεργαστής αναγνωρίζει τα σήματα στάθμης 3.3Volt που στέλνει το XPORT επομένως δεν χρειάζεται περεταίρω αλλαγές πριν σταλούν στον επεξεργαστή. Το αποτέλεσμα της σχεδίασης έχει ως εξής σε επίπεδο σχηματικού:



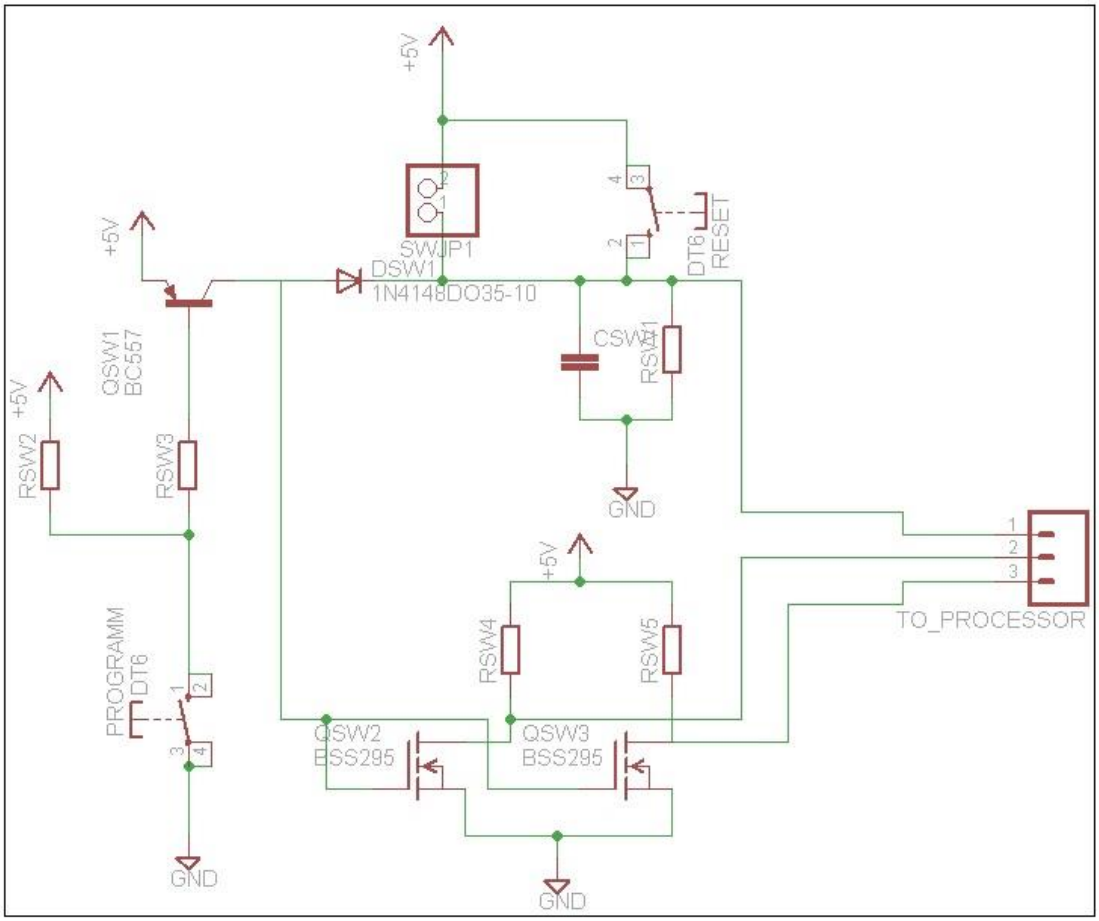
Σχήμα 4.3.1 Συνδεσμολογία XPORT

Για το κεντρικό σύστημα τροφοδοσίας το κύκλωμα χρησιμοποιεί δύο STEP-DOWN τροφοδοτικά τα υπαριθμόν 2575 (ADJ) μεταβλητής τιμής τα οποία προρυθμίζοντε με τους κατάλληλους πυκνωτές και αντιστάσεις να δίνουν τα 5 και 3.3Volt που επιθυμούμαι. Στην σχεδίαση υπάρχει επίσης και η είσοδος 12Volt που παρέχει την τροφοδοσία στο κύκλωμα από εξωτερική πηγή. Για τον έλεγχο του ρεύματος χρησιμοποιούνται δύο πηνία που επιτρέπουν μέγιστο ρεύμα 800mA που είναι και το ρεύμα κόρου αυτών.



Σχήμα 4.3.2: Τροφοδοτικό συστήματος

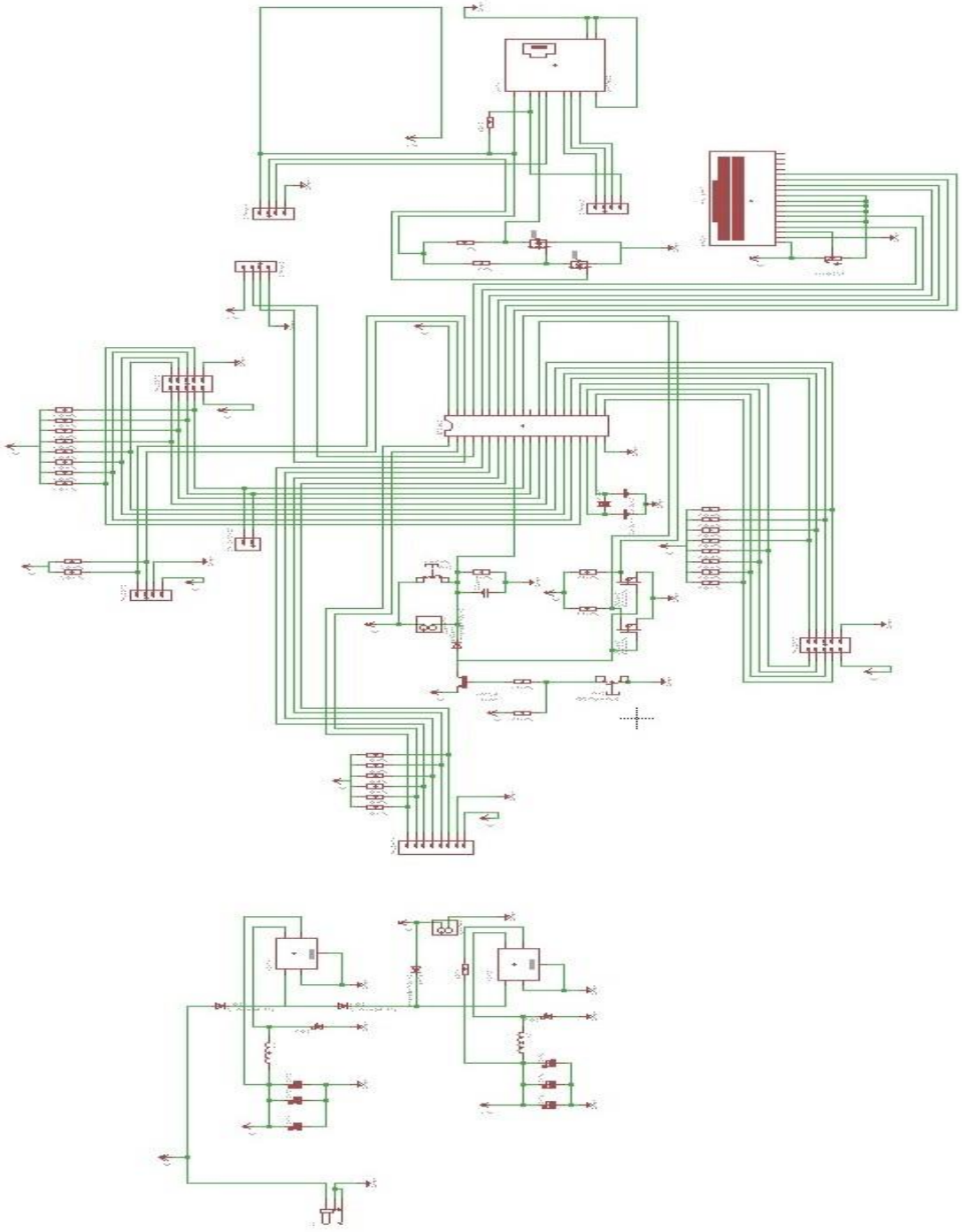
Για ο κύκλωμα μετάβασης του επεξεργαστή σε κατάσταση προγραμματισμού χρησιμοποιούνται δύο Push Button από τα οποία το ένα θέτει τάση στον ακροδέκτη Reset του επεξεργαστή και το άλλο τον βάζει σε κατάσταση προγραμματισμού. Τα σήματα που επεξεργάζεται το συγκεκριμένο σύστημα είναι συνδεδεμένα απευθείας με τους ακροδέκτες του επεξεργαστή για τον έλεγχο του κατά πότε θα μεταβαίνει σε κατάσταση προγραμματισμού αλλά και το πότε θα τίθεται σε κατάσταση λειτουργίας



Σχήμα 4.3.3: Διακόπτης ενεργοποίησης κατάστασης προγραμματισμού επεξεργαστή

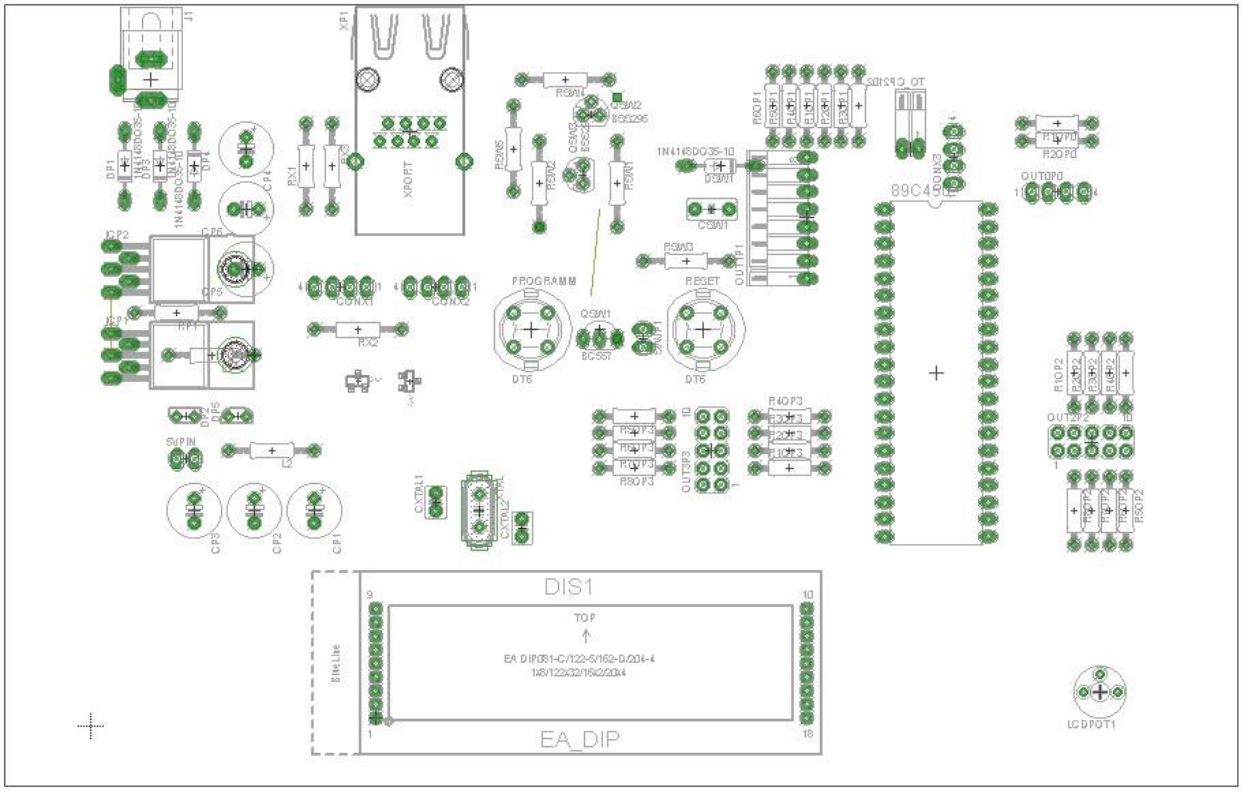
Στο κομμάτι της κεντρικής σχεδίασης υπάρχουν ακροδέκτες εξόδου προς τα περιφερικά, την έξοδο προς την οθόνη LCD για να εμφανίζονται τα μηνύματα κατάστασης , καθώς και τις υπόλοιπες συνδεσμολογίες που συνδέονται με τις πλακέτες που αναλύθηκαν παραπάνω.

Τα σήματα που πηγαίνουν προς τις ακιδοσειρές εξόδου είναι συνδεδεμένα με Pull-Up αντιστάσεις για την ομαλή μετάβαση των σημάτων και αποφυγή πτώσης τάσης σε αυτά.



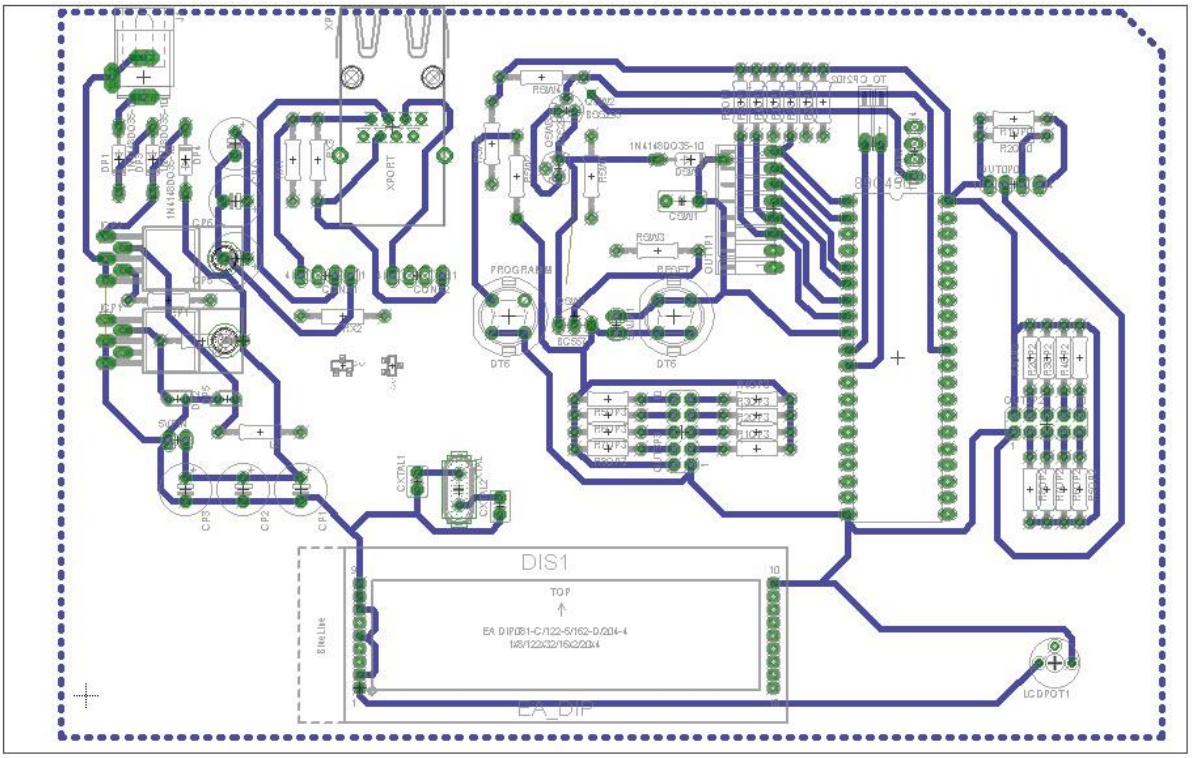
Σχήμα 4.3.4: Ολοκληρωμένο σχεδιάγραμμα πλακέτας

Στο επίπεδο σχεδίασης PCB τα εξαρτήματα τοποθετήθηκαν στις παρακάτω θέσεις όπως δείχνει η εικόνα.



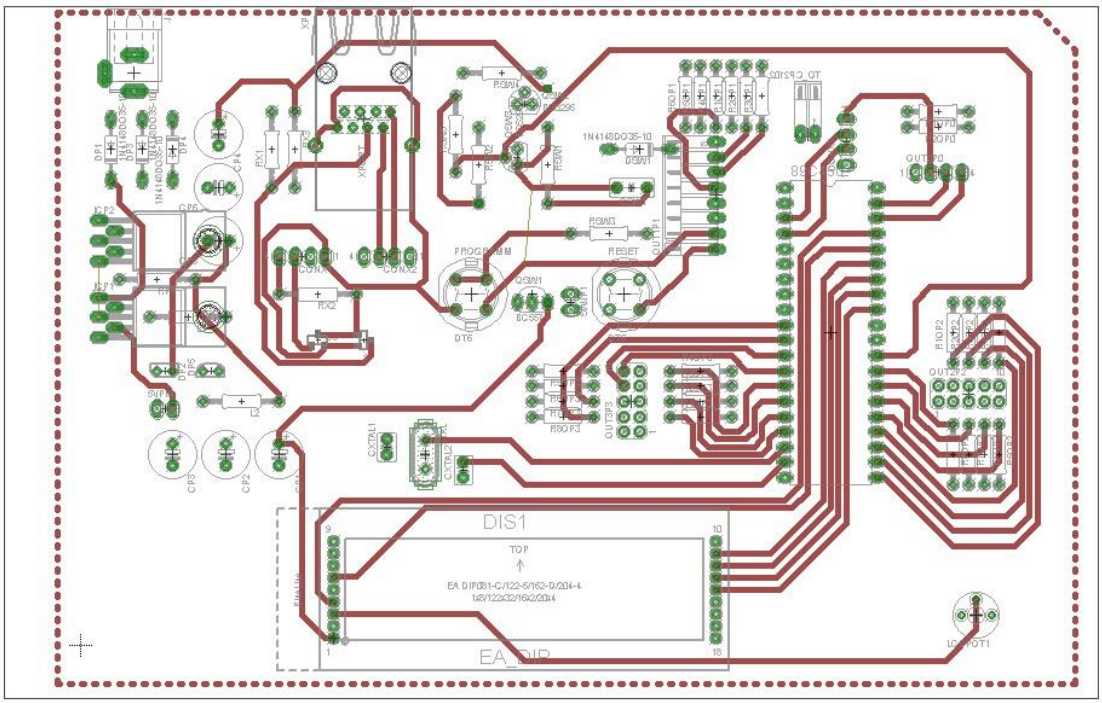
Σχήμα 4.3.5: Τοποθετημένα εξαρτήματα σε επίπεδο πλακέτας

Η πάνω πλευρά της πλακέτας συνδέεται με τις παρακάτω συνδεσμολογίες



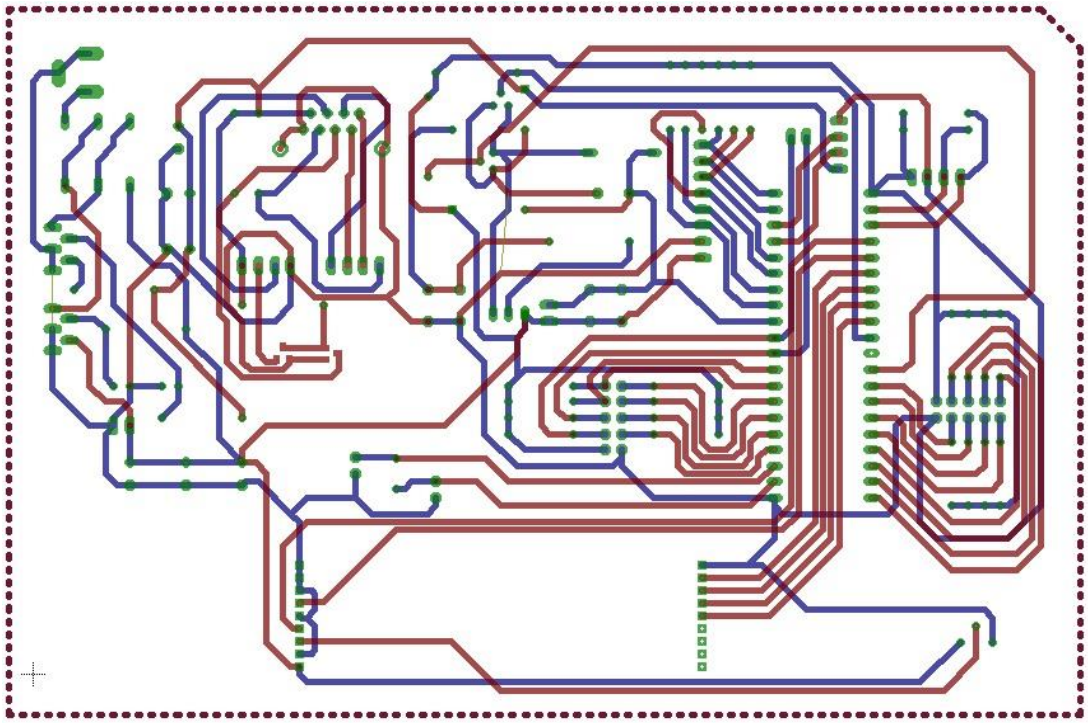
Σχήμα 4.3.6: Συνδεσμολογία Πάνω όψη πλακέτας

Η κάτω πλευρά της πλακέτας συνδέεται με τις συνδεσμολογίες ως εξής:



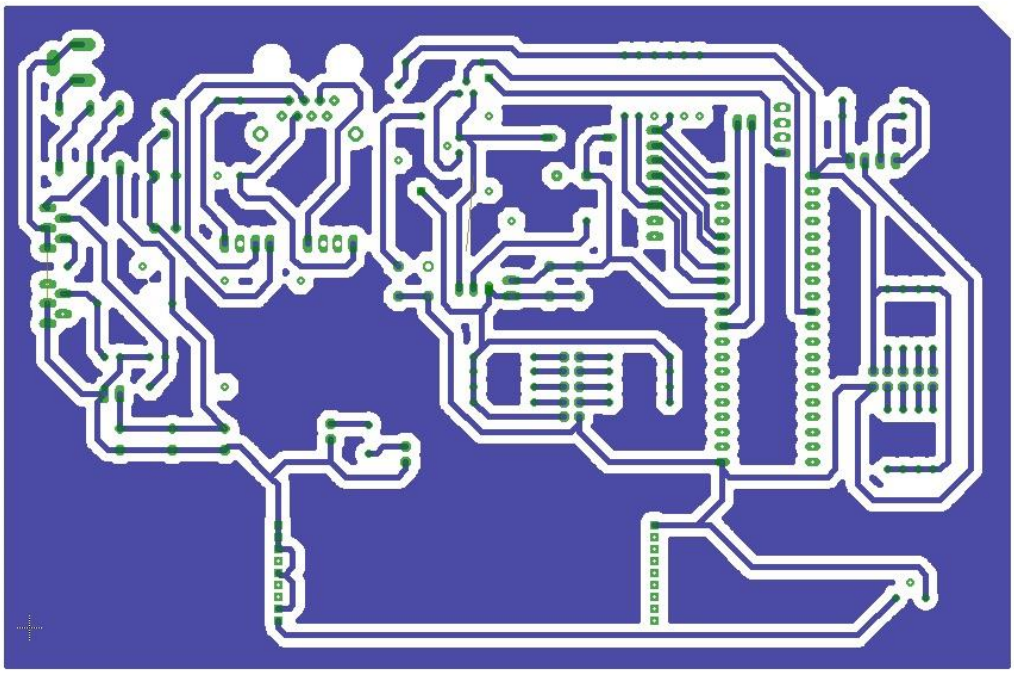
Σχήμα 4.3.7: Συνδεσμολογία Κάτω όψη πλακέτας

Η συνδεσμολογία φαίνεται καλύτερα ως εξής:



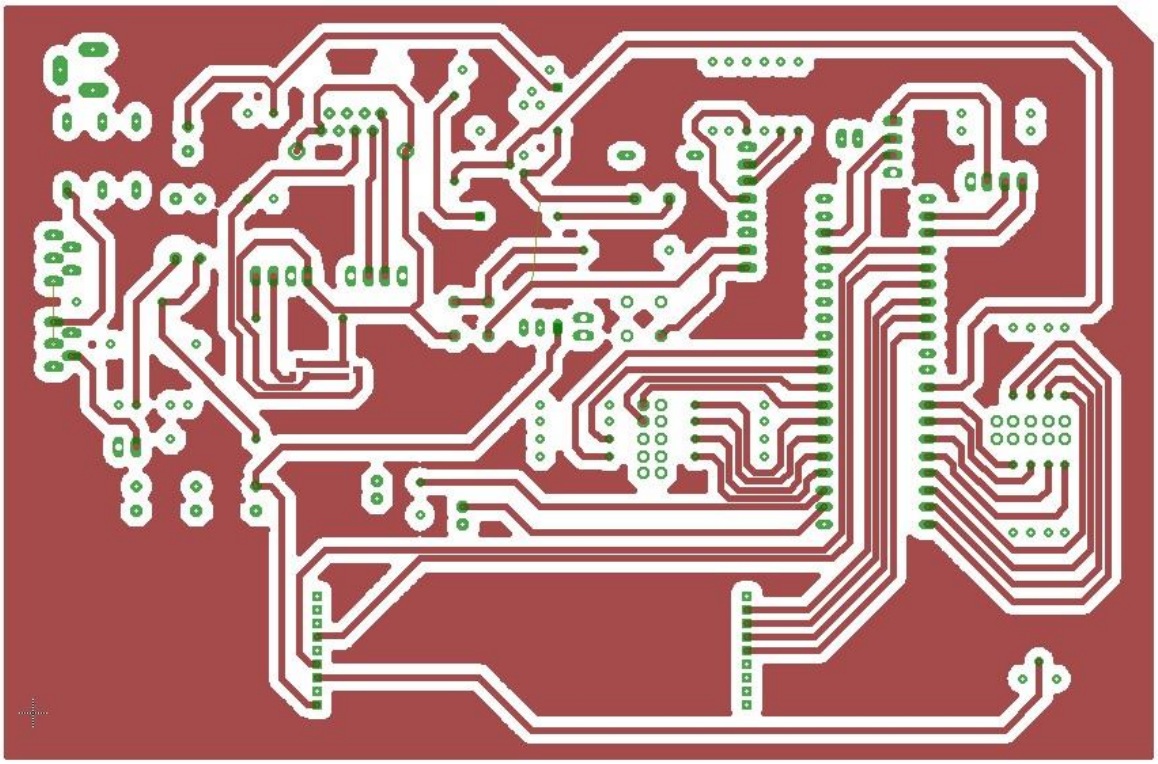
Σχήμα 4.3.8: Συνδυαστικό σχεδιάγραμμα πλακέτας

Αν συμπεριληφθούν και τα πλαίσια γείωσης που φαίνονται παραπάνω θα υπάρξει το εξής αποτέλεσμα τόσο στην πάνω όσο και στην κάτω όψη:
Πάνω όψη:



Σχήμα 4.3.9: Ολοκληρωμένη Πάνω όψη πλακέτας

Κάτω όψη:



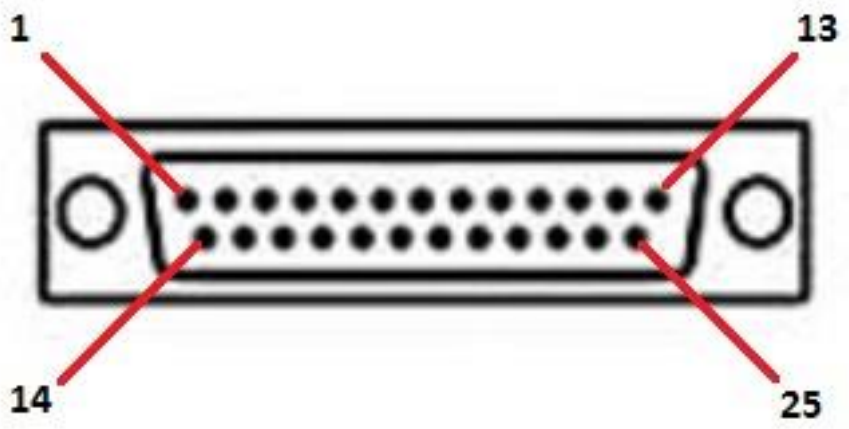
Σχήμα 4.3.9: Ολοκληρωμένη Κάτω όψη πλακέτας

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ/ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

5.1 ΤΟ ΤΕΛΙΚΟ ΣΥΣΤΗΜΑ

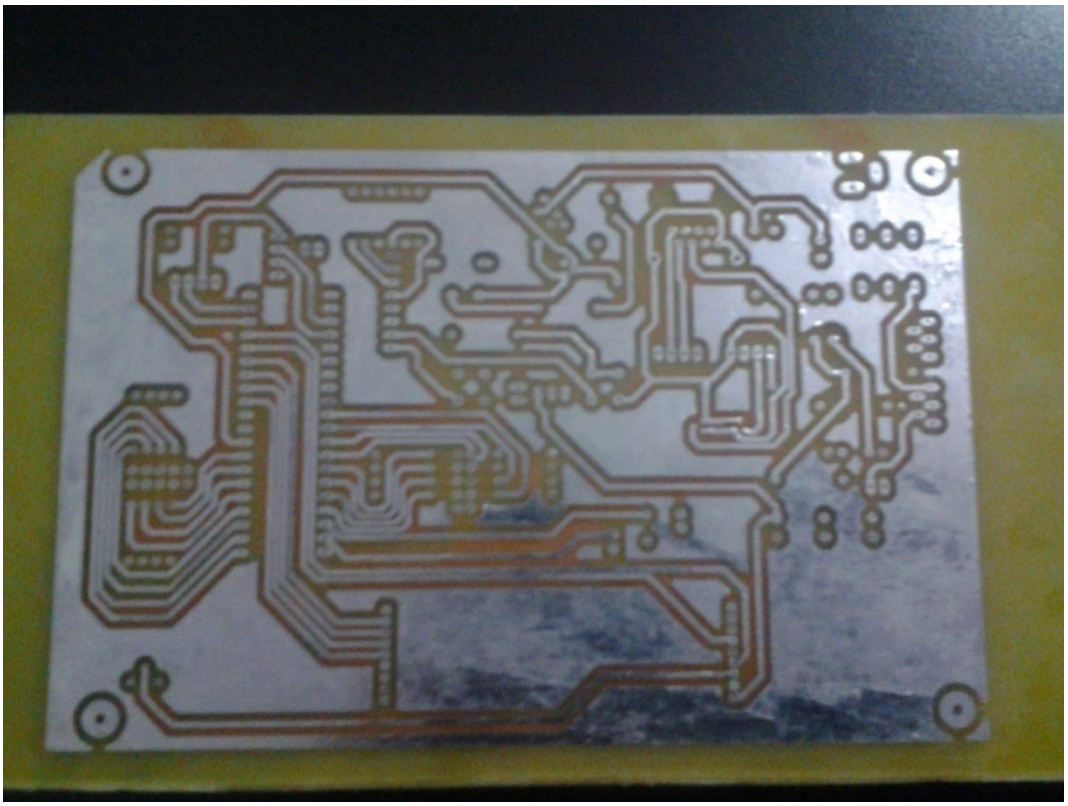
Το τελικό σύστημα όπως παρουσιάστηκε και στα παραπάνω κεφάλαια υλοποιήθηκε παίρνοντας την παρακάτω μορφή. Η κατασκευή τοποθετήθηκε σε βάσεις και προσαρμόστηκε πάνω σε ένα μεταλλικό κουτί το οποίο φέρει υποδοχές για να προεξέχει η τροφοδοσία, είσοδος για το Xport καθώς και για μια σειριακή θύρα 25 ακροδεκτών η οποία είναι σχεδιασμένη να συνδεθεί με τους αισθητήρες με την παρακάτω συνδεσμολογία :



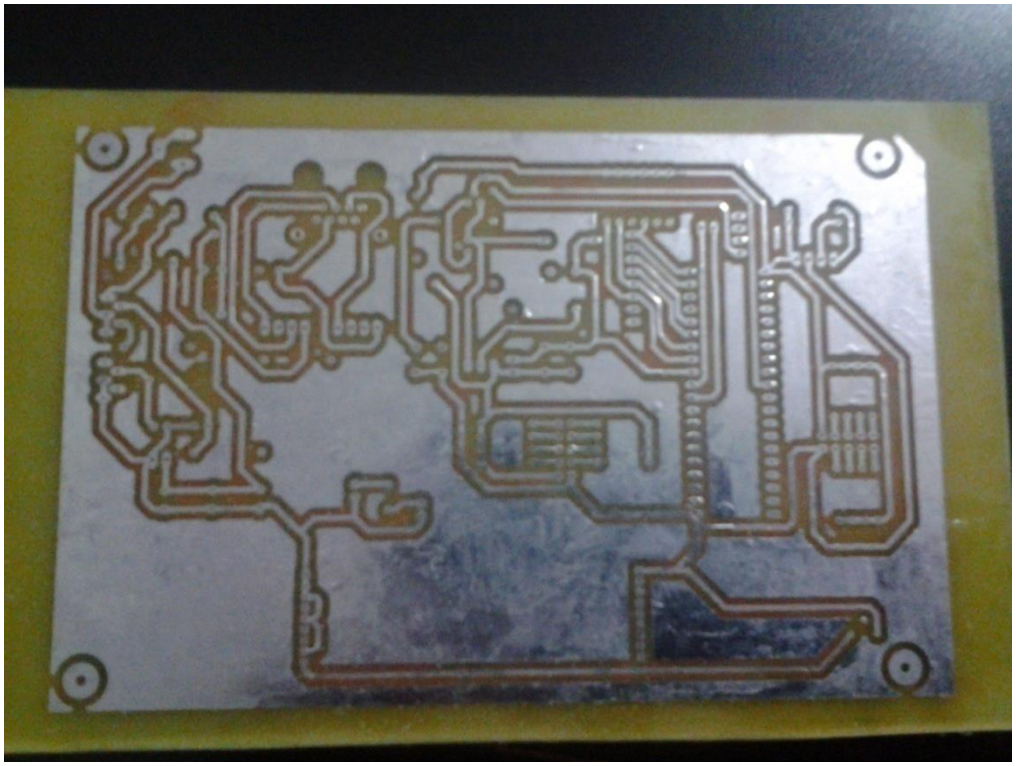
Σχήμα 5.1.1: Ακροδέκτες σειριακής θύρας

Στην θέση 1 έχει τοποθετηθεί ο ακροδέκτης SDA του μικροελεγκτή που υλοποιεί το πρωτόκολλο I2C, στην θέση 14 έχει τοποθετηθεί ο ακροδέκτης SCL του μικροελεγκτή που δίνει τον χρονισμό για την υλοποίηση του παραπάνω πρωτοκόλλου.
Στην θέση 2 είναι τάση τροφοδοσίας 5V που αν και δεν έχει χρησιμοποιηθεί στην κατασκευή επιτρέπει να παρέχεται τροφοδοσία από την κεντρική σχεδίαση στα περιφερικά συστήματα που ελέγχονται.

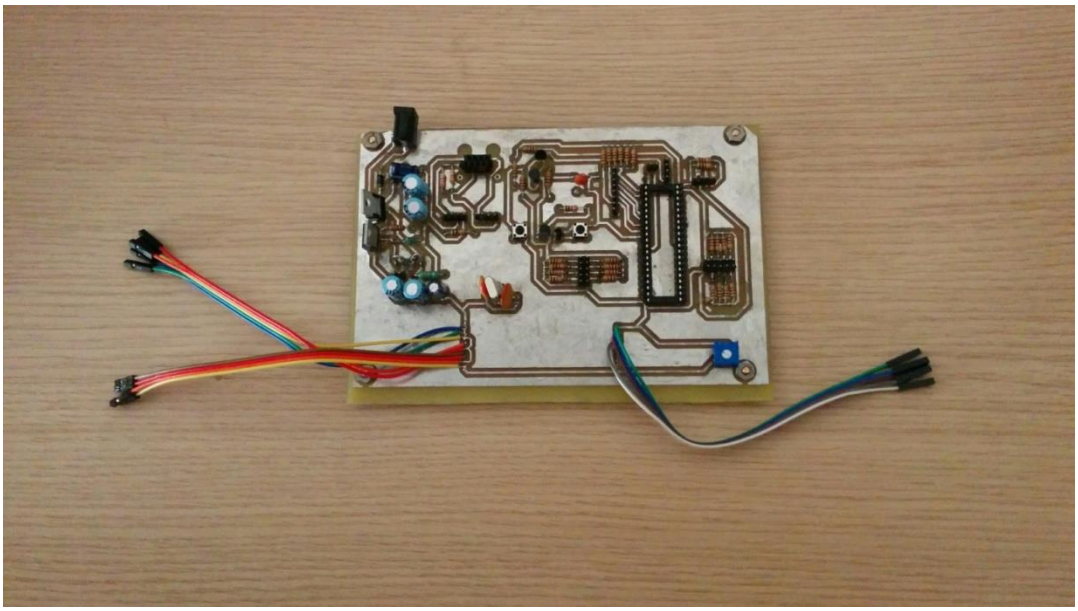
Σε όλη την πάνω ακιδοσειρά έχουν τοποθετηθεί οι επαφές των σημάτων που στέλνουμε και στις αντίστοιχες θέσεις της κάτω ακιδοσειράς έχουν τοποθετηθεί οι ανατροφοδοτήσεις των σημάτων που προέρχονται από τους αισθητήρες προκειμένου να έχει ο χρήστης την δυνατότητα να ελέγχει την σωστή λειτουργία που έχει τεθεί σαν επιθυμητή από το περιβάλλον διασύνδεσης του XPORT.
Από την θέση 3 έως και την θέση 13 οι θέσεις έχουν ως εξής Window1,Lock1,Door1,Lights1, Window2,Lock2,Door2,Lights2, Lock3,Door3,Lights3. Την ίδια σειρά ακολουθούν επομένως και οι κάτω ακροδέκτες με την διαφορά ότι αναφέρονται στα επιστρεφόμενα σήματα από τους αισθητήρες.
Στο επίπεδο της πλακέτας το τελικό αποτέλεσμα είχε ως εξής:



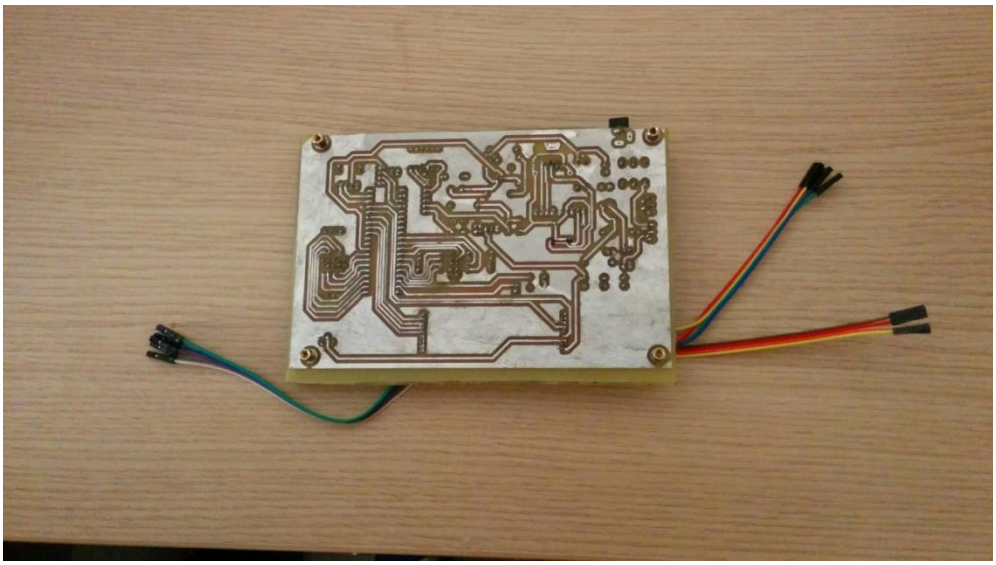
Σχήμα 5.1.2: Κάτω πρόσοψη πλακέτας μετά την επικασσιτέρωση



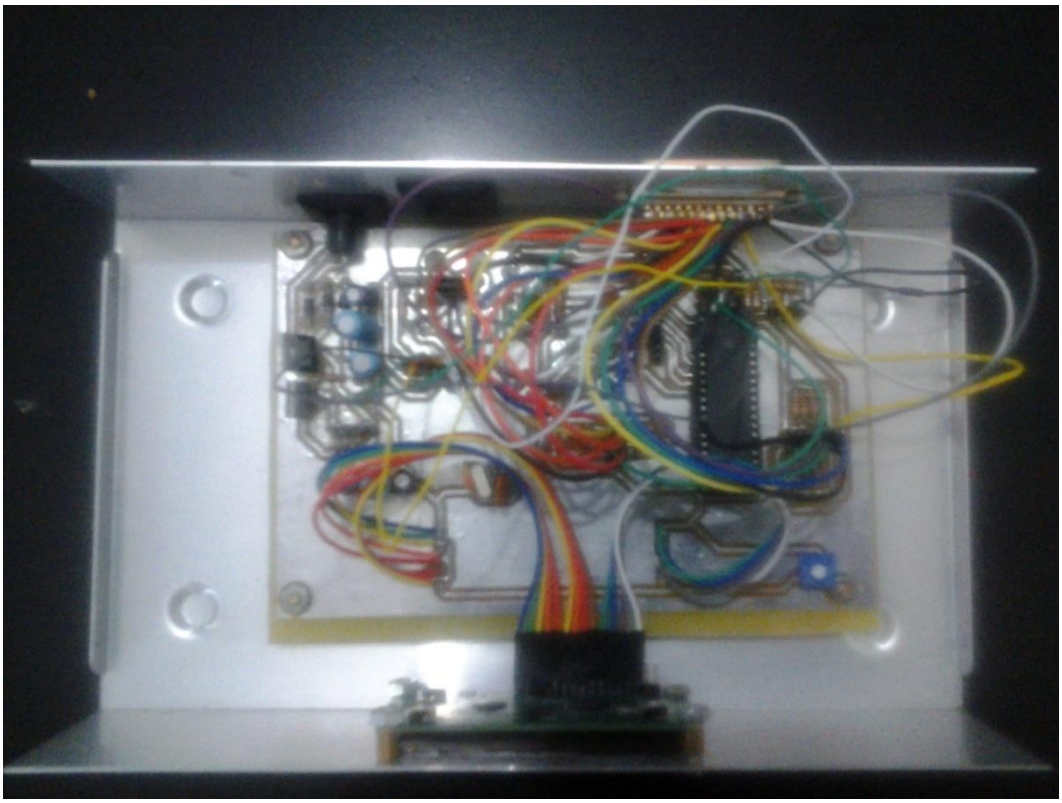
Σχήμα 5.1.3: Πάνω πρόσοψη πλακέτας μετά την επικασσιτέρωση



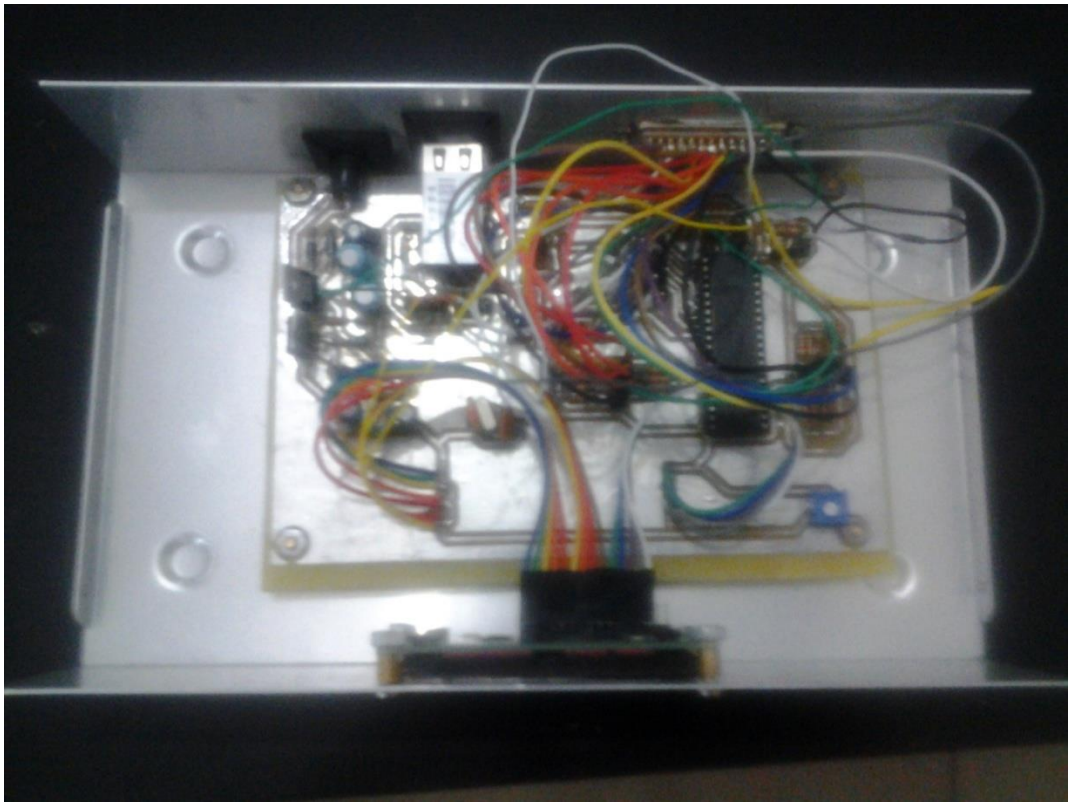
Σχήμα 5.1.4: Πλακέτα με τα βασικά εξαρτήματα χωρίς επεξεργαστή και XPORT



Σχήμα 5.1.5: Κάτω πρόσοψη πλακέτας



Σχήμα 5.1.6: Η τοποθέτηση στο πλαίσιο και η σύνδεση των ακροδεκτών στην LCD καθώς και την σειριακή



Σχήμα 5.1.7: Ολοκλήρωση του συστήματος με προσθήκη επεξεργαστή και XPORT

Όσον αφορά την πρόσοψη του κουτιού που τοποθετήθηκε η πλακέτα υπάρχει η οθόνη LCD η οποία εμφανίζει τα μηνύματα αναλόγως με το εάν οι καταστάσεις στους χώρους επίβλεψης είναι όπως έχουν οριστεί από τον εκάστοτε χρήστη.

Σχήμα 5.1.8: Μπροστινή εμφάνιση πλαισίου

Στην πίσω πλευρά τοποθετήθηκε η σειριακή, η θέση για το Xport και την υποδοχή του Ethernet που θα συνδέεται μαζί του καθώς και η απαραίτητη τροφοδοσία για να λειτουργήσει το κύκλωμα.

Σχήμα 5.1.9: Πίσω όψη πλαισίου

5.2 ΔΥΝΑΤΟΤΗΤΕΣ ΑΝΑΒΑΘΜΙΣΗΣ ΚΑΙ ΕΠΕΚΤΑΣΗΣ

Το σύστημα αυτό στην παρούσα μορφή του χρησιμοποιεί ένα ζευγάρι ακροδεκτών του επεξεργαστή για την υλοποίηση του πρωτοκόλλου I2C ενώ οι υπόλοιπες έξοδοι του λειτουργούν σαν διακόπτες με τροφοδοσία που έχουν άμεση σύνδεση με τους αισθητήρες και τα συστήματα τα οποία ελέγχουν. Πιθανές προεκτάσεις της συγκεκριμένης κατασκευής είναι η μετατροπή όλων των ζευγαριών ακροδεκτών που διαθέτει σε ζευγάρια που υλοποιούν το πρωτόκολλο I2C προκειμένου να πολλαπλασιαστεί ο αριθμός των συσκευών που ελέγχει η σχεδίαση και κατ’επέκταση τον όγκο των συστημάτων που μπορεί να διαχειριστεί. Άλλη εναλλακτική είναι η αλλαγή των εξαρτημάτων που το υποστηρίζουν με εξαρτήματα επιφανειακής στήριξης προκειμένου να ελαττωθεί τον όγκο της πλακέτας. Παρόλο που το σύστημα έχει ήδη αρκετά χαμηλό κόστος για τις υπηρεσίες που παρέχει είναι δυνατόν το κόστος να μειωθεί ακόμα περισσότερο με την επιλογή εξαρτημάτων που εκτελούν διεργασίες περισσότερων του ενός από τα τωρινά εξαρτήματα όπως για παράδειγμα ένα step-down τροφοδοτικό διπλής εξόδου που θα παρείχε τις δύο απαραίτητες τάσεις που χρησιμοποιούνται σε μικρότερο χώρο αλλά και κόστος.

5.3 ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΠΑΡΟΥΣΙΑΣΤΗΚΑΝ

Κατά την διάρκεια της δημιουργίας του συστήματος υπήρξαν προβλήματα που είχαν να κάνουν κατά κύριο λόγο με τον κώδικα ,δεδομένου ότι δεν υπήρχε μεγάλη εξοικείωση με την γλώσσα προγραμματισμού JAVA, τα οποία όμως με την βοήθεια του διαδικτύου αλλά και συναδέλφων που γνώριζαν πώς να υλοποιούν προγράμματα γραμμένα σε αυτήν την γλώσσα ξεπεράστηκαν επιτυχώς. Στην συνέχεια η εύρεση των κατάλληλων εξαρτημάτων για την υλοποίηση του συστήματος δημιούργησε καθυστερήσεις χωρίς όμως να δυσκολέψει ουσιαστικά την διαδικασία. Το βασικό πρόβλημα το οποίο δυστυχώς δεν λύθηκε στα χρονικά περιθώρια που δόθηκαν είναι η μετατροπή του κώδικα σε μορφή αρχείου .cob το οποίο είναι και αυτό που διαβάζει το XPORT. Λόγο αυτού δεν ήταν δυνατόν να πραγματοποιηθεί η επιθυμητή σύνδεση μεταξύ του επεξεργαστή και του XPORT. Όσον αφορά όμως το κομμάτι του κώδικα τόσο σε Java όσο και στην C που χρησιμοποιήθηκε για τον επεξεργαστή γνωρίζουμε ότι είναι λειτουργική λόγο των προγραμμάτων που αναφέρθηκαν παραπάνω και προσομοιώνουν τις συνθήκες λειτουργίας του συστήματος. Στο κομμάτι της κατασκευής είχαμε πολύ μεγάλη βοήθεια από τον υπεύθυνο Κύριο Χατζάκη Ιωάννη καθώς και τον κύριο Ρηγάκη Ηρακλή τόσο για τα εξαρτήματα που θα χρησιμοποιήσουμε όσο και για την δομή που θα πρέπει να έχει το σύστημα μας προκειμένου να είναι λειτουργικό. Το κομμάτι των αισθητήρων δεν θεωρήθηκε απαραίτητο να αγοραστεί δεδομένου ότι δεν θα μπορούσαμε στην παρούσα φάση μέχρι την επίλυση του προβλήματος που είχαμε να αλληλεπιδράσει η σχεδίαση μας με φυσικά στοιχεία που θα συνδέονταν στις εξόδους της.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1]IEEE Transactions on Automation Science and Engineering
- [1]PLC Control Panels
- [1]Jobs in Automation, Robotics and Process Control
- [1]ISA (International Society of Automation)
- [1]Intel's Automation Process and Its Role in Process Development and High Volume Manufacturing

- [2]Η Ελλάδα στο Διαδίκτυο
- [2]*World of Ends, What the Internet Is and How to Stop Mistaking It for Something Else* by Doc Searls and David Weinberger
- [2]The Internet Society (ISOC)
- [2]Internet Mapping Project
- [2]RFC 801, planning the TCP/IP switchover
- [2]V7ndotcom Elursrebmem
- [2]Access at home, by native language
- [2]John Walker: The Digital Imprimatur
- [2]Εισαγωγή στην προσβασιμότητα στο διαδίκτυο
- [3]Gosling, James, *A brief history of the Green project*.[dead link] Java.net, no date [ca. Q1/1998]. Retrieved April 29, 2007.
- [3]Gosling, James, *A brief history of the Green project*.[dead link] anonymous-insider.net, no date [ca. Q1/1998]. Retrieved September 4, 2013.
- [3]Gosling, James; Joy, Bill; Steele, Guy L., Jr.; Bracha, Gilad (2005). *The Java Language Specification* (3rd ed.). Addison-Wesley. ISBN 0-321-24678-0.
- [3]Lindholm, Tim; Yellin, Frank (1999). *The Java Virtual Machine Specification* (2nd ed.). Addison-Wesley. ISBN 0-201-43294-3.
-
- [4]Official I2C Specification, NXP
- [4]Δικτυακά Σεμινάρια προχωρημένης Ανάλυσης του Διαύλου I²C
- [4]I²C Υπόβαθρο
- [4]I²C Bus / Access Bus
- [4]Πιστοποιημένες πραγματώσεις του Διαύλου I²C

- [4]OpenBSD iic(4) manual page
- [4]massmind I²C page Source code, samples and technical information for using I²C with PC, PIC and SX microcontrollers.
- [4]Σελίδα Πληροφοριών Σειριακών Διαύλων
- [4]εχνική περίληψη του διαύλου I²C και συχνές ερωτήσεις
- [4]The Bus Buffer Resource. For 2-wire buses such as I²C, SMBus, PMBus, IPMB & IPMI
- [4]I²C Protocol
- [4]I²C logic microchips from Texas Instruments
- [4]Εισαγωγή στο δίαυλο I²C
- [4]Εισαγωγή στα πρωτόκολλα SPI και I2C
- [4]Κώδικας Αποκωδικοποίησης Πρωτοκόλου I2C
- [4]Beginner's guide to using Arduino with I²C devices, including worked examples
- [4]Επιπτώσεις μεταβολής των Pull-up αντιστάσεων στον δίαυλο I²C
- [5]http://www.nesweb.ch/downloads/XPort.pdf
- [6]http://datasheets.maximintegrated.com/en/ds/DS89C430-DS89C450.pdf
- [7]http://www.mikroe.com/chapters/view/69/chapter-6-examples/
- [7]http://naveenauvusali.blogspot.gr/p/io-interfacing.html
- [7]http://www.edn.com/design/analog/4371297/Design-calculations-for-robust-I2C-communications
- [7]http://henrypoon.wordpress.com/2011/01/01/serial-communication-in-java-with-example-program/
- [7]http://www.homeandlearn.co.uk/java/java_radio_buttons.html
- [7]http://www.java-samples.com/showtutorial.php?tutorialid=214
- [7]http://www.w3schools.com/default.asp
- [7]http://www.codecademy.com/learn
- [8]Official website
- [8]IBM Rational and Eclipse
- [8]Eclipse 4 RCP Tutorial Tutorial for Eclipse 4
- [8]Using Eclipse as a Front-End to GDB Debugger
- [9] http://www.keil.com/support/man/docs/uv4/uv4_examples.htm
- [9] http://www.keil.com/uvision/ide_ov_examples.asp
- [9]SDCC

- [9]Keil C
- [9]IAR Systems C/C++
- [9]MikroElektronika C
- [9]Tasking C
- [9]Bascom51 Basic compiler
- [9]turbo51 Pascal
- [9]Forth
- [9] http://www.keil.com/uvision/

ΠΑΡΑΡΤΗΜΑ

- **DC Χαρακτηριστικά για σειριακή θύρα, Pin εισόδου/εξόδου, και τροφοδοσία για το XPORT-05R**

Σύμβολο	Παράμετρος	Ελάχιστη Τιμή	Κανονική Τιμή	Μέγιστη Τιμή	Μονάδες
Vcc	Τάση τροφοδοσίας (τυπική 3.3) (+/-5%)	3.14	3.3	3.46	V
VIL	Χαμηλή στάθμη Τάσης Εισόδου	0		0.8	V
VIH	Υψηλή στάθμη Τάσης Εισόδου	2.0		5.5	V
VOL	Χαμηλή στάθμη Τάσης Εξόδου			0.4	V
VOH	Υψηλή στάθμη Τάσης Εξόδου	2.4			V
II	Ρεύμα διαρροής εισόδου			1	μA
ICC	Ρεύμα τροφοδοσίας (σε αδράνεια)@ 48 MHz		119		mA
ICC	Ρεύμα τροφοδοσίας (10BASE-T Δραστηριοτητα)@ 48 MHz		224		mA
ICC	Ρεύμα τροφοδοσίας (10BASE-T Δραστηριοτητα)@ 88 MHz		267		mA
ICC	Ρεύμα τροφοδοσίας (100BASE-T Δραστηριοτητα)@ 48 MHz		190		mA
ICC	Ρεύμα τροφοδοσίας (100BASE-T Δραστηριοτητα)@ 88 MHz		233		mA

❖ DC και AC Χαρακτηριστικά DS89C450:

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V, T_O = -40°C to +85°C.) (Note 1)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
Supply Voltage (Notes 2, 3)	V_{CC}	4.5	5.0	5.5	V
Power-Fail Warning (Notes 2, 4)	V_{PFW}	4.2	4.375	4.6	V
Reset Trip Point (Min Operating Voltage) (Notes 2, 3, 4)	V_{RST}	3.95	4.125	4.35	V
Supply Current, Active Mode (Note 5)	I_{CC}		75	110	mA
Supply Current, Idle Mode at 33MHz (Note 6)	I_{IDLE}		40	50	mA
Supply Current, Stop Mode, Bandgap Disabled (Note 7)	I_{STOP}		1	100	µA
Supply Current, Stop Mode, Bandgap Enabled (Note 7)	I_{SPBG}		150	300	µA
Input Low Level (Note 2)	V_{IL}	-0.3		+0.8	V
Input High Level (Note 2)	V_{IH}	2.0		V_{CC} + 0.3	V
Input High Level XTAL and RST (Note 2)	V_{IH2}	3.5		V_{CC} + 0.3	V
Output Low Voltage, Port 1 and 3 at I_{OL} = 1.6mA (Note 2)	V_{OL1}		0.15	0.45	V
Output Low Voltage, Port 0 and 2, ALE, $\overline{PSEN}$ at I_{OL} = 3.2mA (Note 2)	V_{OL2}		0.15	0.45	V
Output High Voltage, Port 1, 2, and 3, at I_{OH} = -50µA (Notes 2, 8)	V_{OH1}	2.4			V
Output High Voltage, Port 1, 2, and 3 at I_{OH} = -1.5mA (Notes 2, 9)	V_{OH2}	2.4			V
Output High Voltage, Port 0, 1, 2, ALE, $\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$ in Bus Mode at I_{OH} = -8mA (Notes 2, 10)	V_{OH3}	2.4			V
Output High Voltage, RST at I_{OL} = -0.4mA (Note 2, 11)	V_{OH4}	2.4			V
Input Low Current, Port 1, 2, and 3 at 0.4V	I_{IL}	-50			µA
Transition Current from 1 to 0, Port 1, 2, and 3 at 2V (Note 12)	I_{TL}	-650			µA
Input Leakage Current, Port 0 in I/O Mode and $\overline{EA}$ (Note 13)	I_L	-10		+10	µA
Input Current, Port 0 in Bus Mode (Note 14)	I_L	-300		+300	µA
RST Pulldown Resistance (Note 13)	R_{RST}	50	120	200	kΩ

AC CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V, T_O = -40°C to +85°C.) (See Figure 1, Figure 2, and Figure 3.)

PARAMETER	SYMBOL	1-CYCLE PAGE MODE 1		2-CYCLE PAGE MODE 1		4-CYCLE PAGE MODE 1		PAGE MODE 2		NONPAGE MODE		UNITS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
System Clock External Oscillator (Note 15)	$1/t_{CLCL}$	0	33	0	33	0	33	0	33	0	33	MHz
System Clock External Crystal (Note 15)	$1/t_{CLCL}$	1	33	1	33	1	33	1	33	1	33	
ALE Pulse Width (Note 16)	t_{LHLL}	$0.5t_{CLCL}$ - 2 + t_{STC3}		t_{CLCL} - 2 + t_{STC3}		$2t_{CLCL}$ - 4 + t_{STC3}		$1.5t_{CLCL}$ - 5 + t_{STC3}		$1.5t_{CLCL}$ - 5 + t_{STC3}		ns
Port 0 Instruction Address Valid to ALE Low	t_{AVLL}							t_{CLCL} - 3		$0.5t_{CLCL}$ - 3		ns
Port 2 Instruction Address Valid to ALE Low	t_{AVLL2}	$0.5t_{CLCL}$ - 4		$0.5t_{CLCL}$ - 4		$1.5t_{CLCL}$ - 4		$0.5t_{CLCL}$ - 4		t_{CLCL} - 4		ns
Port 0 Data AddressValid to ALE Low	t_{AVLL3}							t_{CLCL} - 3 + t_{STC3}		$0.5t_{CLCL}$ - 3 + t_{STC3}		ns
Program Address Hold After ALE Low	t_{LLAX}	$0.5t_{CLCL}$ - 8		$1.5t_{CLCL}$ - 8		$2.5t_{CLCL}$ - 8		$1t_{CLCL}$ - 10		$1t_{CLCL}$ - 10		ns
Address Hold after ALE Low MOVX Write	t_{LLAX2}	$0.5t_{CLCL}$ - 8 + t_{STC4}		$1.5t_{CLCL}$ - 8 + t_{STC4}		$2.5t_{CLCL}$ - 8 + t_{STC3}		$0.5t_{CLCL}$ - 8 + t_{STC2}		$0.5t_{CLCL}$ - 8 + t_{STC2}		ns
Address Hold after ALE Low MOVX Read	t_{LLAX3}	$0.5t_{CLCL}$ - 8 + t_{STC4}		$1.5t_{CLCL}$ - 8 + t_{STC4}		$2.5t_{CLCL}$ - 8 + t_{STC3}		$0.5t_{CLCL}$ - 8 + t_{STC3}		$0.5t_{CLCL}$ - 8 + t_{STC2}		ns
ALE Low to Valid Instruction In	t_{LLIV}								$2t_{CLCL}$ - 6		$2t_{CLCL}$ - 6	ns
ALE Low to $\overline{PSEN}$ Low	t_{LLPL}							$1.5t_{CLCL}$ - 6		$0.5t_{CLCL}$ - 2		ns
$\overline{PSEN}$ Pulse Width for Program Fetch	t_{PLPH}	t_{CLCL} - 5		t_{CLCL} - 5		$2t_{CLCL}$ - 5		t_{CLCL} - 5		$2t_{CLCL}$ - 5		ns

AC CHARACTERISTICS (continued)

(V_{CC} = 4.5V to 5.5V, T_O = -40°C to +85°C.) (See Figure 1, Figure 2, and Figure 3.)

PARAMETER	SYMBOL	1-CYCLE PAGE MODE 1		2-CYCLE PAGE MODE 1		4-CYCLE PAGE MODE 1		PAGE MODE 2		NONPAGE MODE		UNITS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$\overline{PSEN}$ Low to Valid Instruction In	t_{PLIV}		t_{CLCL} - 20		t_{CLCL} - 20		$2t_{CLCL}$ - 20		t_{CLCL} - 20		$2t_{CLCL}$ - 20	ns
Input Instruction Hold After $\overline{PSEN}$	t_{PXIX}	0		0		0		0		0		ns
Input Instruction Float After $\overline{PSEN}$	t_{PXIZ}								t_{CLCL} - 5		t_{CLCL} - 5	ns
Port 0 Address to Valid Instruction In	t_{AVIV0}								$1.5t_{CLCL}$ - 22		$3t_{CLCL}$ - 22	ns
Port 2 Address to Valid Instruction In	t_{AVIV2}		t_{CLCL} - 20		$1.5t_{CLCL}$ - 20		$2.5t_{CLCL}$ - 20		$3t_{CLCL}$ - 20		$3.5t_{CLCL}$ - 20	ns
$\overline{PSEN}$ Low to Port 0 Address Float	t_{PLAZ}								0		0	ns
$\overline{RD}$ Pulse Width (P3.7) (Note 16)	t_{RLRH}	t_{CLCL} - 5 + t_{STC1}		t_{CLCL} - 5 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}		ns
$\overline{WR}$ Pulse Width (P3.6) (Note 16)	t_{WLWH}	t_{CLCL} - 5 + t_{STC1}		t_{CLCL} - 5 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}		ns
$\overline{RD}$ (P3.7) Low to Valid Data In (Note 16)	t_{RLDV}		t_{CLCL} - 18 + t_{STC1}		t_{CLCL} - 18 + t_{STC1}		$2t_{CLCL}$ - 18 + t_{STC1}		$2t_{CLCL}$ - 18 + t_{STC1}		$2t_{CLCL}$ - 18 + t_{STC1}	ns
Data Hold After $\overline{RD}$ (P3.7)	t_{RHDX}	0		0		0		0		0		ns
Data Float After $\overline{RD}$ (P3.7)	t_{RHDZ}								t_{CLCL} - 5		t_{CLCL} - 5	ns
MOVX ALE Low to Input Data Valid (Note 16)	t_{LLDV}								$2t_{CLCL}$ - 8 + t_{STC1}		$2t_{CLCL}$ - 5 + t_{STC1}	ns

AC CHARACTERISTICS (continued)

(V_{CC} = 4.5V to 5.5V, T_O = -40°C to +85°C.) (See Figure 1, Figure 2, and Figure 3.)

PARAMETER	SYMBOL	1-CYCLE PAGE MODE 1		2-CYCLE PAGE MODE 1		4-CYCLE PAGE MODE 1		PAGE MODE 2		NONPAGE MODE		UNITS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
Port 0 Address to Valid Data In (Note 16)	t_{AVDV0}								$3t_{CLCL}$ - 20 + t_{STC1}		$3t_{CLCL}$ - 20 + t_{STC1}	ns
Port 2 Address to Valid Data In (Note 16)	t_{AVDV2}		t_{CLCL} - 20 + t_{STC1}		$1.5t_{CLCL}$ - 20 + t_{STC1}		$3.5t_{CLCL}$ - 20 + t_{STC1}		$3.0t_{CLCL}$ - 20 + t_{STC1}		$3.5t_{CLCL}$ - 20 + t_{STC1}	ns
ALE Low to $\overline{RD}$ or $\overline{WR}$ Low (Note 16)	t_{LLRL} (t_{LLWL})	$0.5t_{CLCL}$ - 8 + t_{STC2}	$0.5t_{CLCL}$ + 6 + t_{STC2}	$2t_{CLCL}$ - 8 + t_{STC2}	$2t_{CLCL}$ + 6 + t_{STC2}	$4t_{CLCL}$ - 8 + t_{STC2}	$4t_{CLCL}$ + 6 + t_{STC2}	$0.5t_{CLCL}$ - 8 + t_{STC2}	$0.5t_{CLCL}$ + 4 + t_{STC2}	$0.5t_{CLCL}$ - 8 + t_{STC2}	$0.5t_{CLCL}$ + 5 + t_{STC2}	ns
Port 0 Address Valid to $\overline{RD}$ or $\overline{WR}$ Low (Note 16)	t_{AVRL0} (t_{AVWL0})							$1.5t_{CLCL}$ - 5 + t_{STC2}		t_{CLCL} - 5 + t_{STC2}		ns
Port 2 Address Valid to $\overline{RD}$ or $\overline{WR}$ Low (Note 16)	t_{AVRL2} (t_{AVWL2})	0 + t_{STC5} - 5		$0.5t_{CLCL}$ - 5 + t_{STC5}		$1.5t_{CLCL}$ - 5 + t_{STC5}		t_{CLCL} - 5 + t_{STC5}		$1.5t_{CLCL}$ - 5 + t_{STC5}		ns
Data Out Valid to $\overline{WR}$ Transition (Note 15)	t_{QVWX}	-5		-5		-5		-5		-5		ns
Data Hold After $\overline{WR}$ (Note 15)	t_{WHQX}	t_{CLCL} + t_{STC2} - 10		t_{CLCL} + t_{STC2} - 10		t_{CLCL} + t_{STC2} - 10		t_{CLCL} + t_{STC2} - 10		t_{CLCL} + t_{STC2} - 10		ns
$\overline{RD}$ or $\overline{WR}$ High to ALE High (Note 15)	t_{RHLH} (t_{WHLH})	t_{STC2} - 2	t_{STC2} + 4	t_{STC2} - 2	t_{STC2} + 4	t_{STC2} - 2	t_{STC2} + 4	t_{STC2} - 2	t_{STC2} + 4	t_{STC2} - 2	t_{STC2} + 4	ns

Note: Specifications to -40°C are guaranteed by design and are not production tested. AC electrical characteristics assume 50% duty cycle for the oscillator and are not 100% tested, but are guaranteed by design.

EXTERNAL CLOCK CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V, T_O = -40°C to +85°C.)

PARAMETER	SYMBOL	MIN	MAX	UNITS
Clock High Time	t_{CHCX}	10		ns
Clock Low Time	t_{CLCX}	10		ns
Clock Rise Time	t_{CLCH}		5	ns
Clock Fall Time	t_{CHCL}		5	ns

SERIAL PORT MODE 0 TIMING CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V, T_O = -40°C to +85°C.) (Figure 4)

PARAMETER	SYMBOL	CONDITIONS	33MHz		VARIABLE		UNITS
			MIN	MAX	MIN	MAX	
Clock Cycle Time	t_{XLXL}	SM2 = 0	360		$12t_{CLCL}$		ns
		SM2 = 1	120		$4t_{CLCL}$		ns
Output Data Setup to Clock Rising	t_{QVXH}	SM2 = 0	200		$10t_{CLCL}$ - 100		ns
		SM2 = 1	40		$3t_{CLCL}$ - 10		ns
Output Data Hold to Clock Rising	t_{XHQX}	SM2 = 0	50		$2t_{CLCL}$ - 10		ns
		SM2 = 1	20		t_{CLCL} - 100		
Input Data Hold After Clock Rising	t_{XHDX}	SM2 = 0	0		0		ns
		SM2 = 1	0		0		
Clock Rising Edge to Input Data Valid	t_{XHDV}	SM2 = 0		200		$10t_{CLCL}$ - 100	ns
		SM2 = 1		40		$3t_{CLCL}$ - 50	ns

Note: SM2 is the serial port 0 mode bit 2. When serial port 0 is operating in mode 0 (SM0 = SM1 = 0), SM2 determines the number of crystal clocks in a serial port clock cycle.

Figure 4. Serial Port Timing

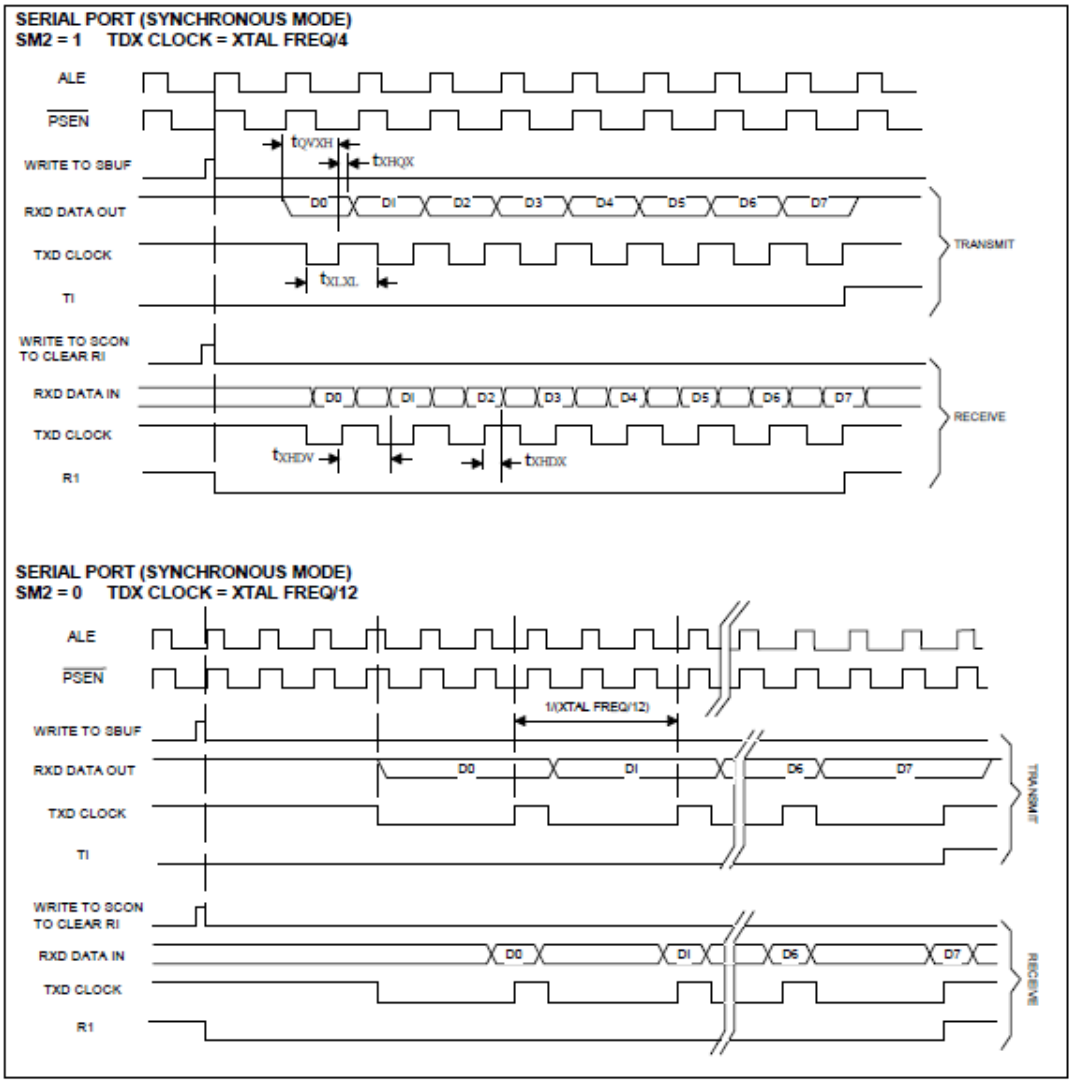


Table 1. SFR Register Map

REGISTER	ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
P0	80h	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
SP	81h								
DPL	82h								
DPH	83h								
DPL1	84h								
DPH1	85h								
DPS	86h	ID1	ID0	TSL	AID	—	—	—	SEL
PCON	87h	SMOD_0	SMOD0	OFDF	OFDE	GF1	GF0	STOP	IDLE
TCON	88h	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TMOD	89h	GATE	C/T	M1	M0	GATE	C/T	M1	M0
TL0	8Ah								
TL1	8Bh								
TH0	8Ch								

❖ Κώδικας Περιβάλλοντος JAVA

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JToggleButton;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;

public class panel_instance() {
   zone=1;
   windows=false;
   doors=false;
   locks=false;
   lights=false;
   code={a,a,0}
}

// the Bicycle class has
// four methods
public void setZone(int newZone) {
   zone = newZone;
   setCode(zone,code[1],code[2])
}
```

```
public void setWindow(boolean newWin) {
   windows = newWin;
   if (windows==true)
      {setCode(code[0],"a","1")}
      else setCode(code[0],"a","0");
}

public void setDoors(boolean newDoor) {
   doors = newDoor;
if (doors==true)
      {setCode(code[0],"b","1")}
      else setCode(code[0],"b","0");
}

public void setLock(boolean newLock) {
   locks = newLock;
   if (locks==true)
      {setCode(code[0],"c","1")}
      else setCode(code[0],"c","0");
}

public void setLights(boolean newLight) {
   lights = newLight;
   if (lights==true)
      {setCode(code[0],"d","1")}
      else setCode(code[0],"d","0");
}

public void setCode(char a , char b ,char c) {
   code[0]=a;
   code[1]=b;
   code[2]=c;
```

```
}
}

   public class example extends JFrame
   {
      public example() { initUI();

                  }


      public final void initUI()
         {
            JPanel panel = new JPanel();
            getContentPane().add(panel);
            panel.setLayout(null);
            panel_instance panel1 = new panel_instance();

            JToggleButton windowsButton = new JToggleButton("Windows");
            windowsButton.setBounds(150, 100, 120, 30);
            windowsButton.
            windowsbutton.addActionListener(new ActionListener() {

               public void actionPerformed(ActionEvent e)
               {
                  //Execute when button is pressed
                  if(panel1.windows=="false")
                  {panel1.setWindow(true);}
                  else panel1.setWindow(false);
               }
            });

            JToggleButton doorsButton = new JToggleButton("Doors");
            doorsButton.setBounds(150, 30, 120, 30);
```

```
doors
button.addActionListener(new ActionListener() {

   public void actionPerformed(ActionEvent e)
   {
      //Execute when button is pressed
      if(panel1.doors=="false")
      {panel1.setdoors(true);}
      else panel1.setdoors(false);
   }
});
panel.add(doorsButton);

JToggleButton locksButton = new JToggleButton("Locks");
locksButton.setBounds(20, 30, 120, 30);
locksbutton.addActionListener(new ActionListener() {

   public void actionPerformed(ActionEvent e)
   {
      //Execute when button is pressed
      if(panel1.locks=="false")
      {panel1.setlocks(true);}
      else panel1.setlocks(false);
   }
});
panel.add(locksButton);

JToggleButton lightsButton = new JToggleButton("Lights");
lightsButton.setBounds(20, 100, 120, 30);
lights
button.addActionListener(new ActionListener() {

   public void actionPerformed(ActionEvent e)
```

```
    {
        //Execute when button is pressed
        if(panel1.lights=="false")
        {panel1.setlights(true);}
        else panel1.setlights(false);
    }
});
panel.add(lightsButton);

JRadioButton zone1= new JRadioButton("zone1");
zone1.setBounds(20, 2, 60, 20);
zone1.addActionListener(null);
panel.add(zone1);

JRadioButton zone2= new JRadioButton("zone2");
zone2.setBounds(80, 2, 60, 20);
zone2.addActionListener(null);
panel.add(zone2);

JRadioButton zone3= new JRadioButton("zone3");
zone3.setBounds(140, 2, 60, 20);
zone3.addActionListener(null);
panel.add(zone3);

JRadioButton zone4= new JRadioButton("zone4");
zone4.setBounds(200, 2, 60, 20);
zone4.addActionListener(null);
panel.add(zone4);

    ButtonGroup zone = new ButtonGroup( );

    zone.add(zone1);
    zone.add(zone2);
```

```
            zone.add(zone3);
            zone.add(zone4);

        setTitle("User Interface");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
      }

   public static void main(String[] args)
   {
      example ex = new example(); ex.setVisible(true);

   }
}
```

- **Κώδικας για προγραμματισμό του επεξεργαστή**

```
////////////////////////////////////////////////CPU CODE/////////////////////////////////////////////

#include <DS89C4XX.h>    //DS89C450 Microcontroller
#include <intrins.h>    //_nop_() function
#include <stdio.h>        //sprintf() function
#define  _Nop()  _nop_()
#include <string.h>
#include <i2c.h>
#include <lcd.h>

//////////////////////////////////////delay ////////////////////////////////////////////////
void delay(int count)
{
   int i,j;

   for (i=0;i<count;i++){
   for (j=0;j<1200;j++); }
}
//-------------------------------------------------

//////////////////////////////LCD Initialisation//////////////////////////////////////////////
void norm_show(void)
{
 lcd_clear();

 printf("\n\r\n\rSYSTEM OPERATING\n\r");
}
/////////////////////////////////////////////////////////////////////////////////////////////

sbit SDA=P0^1;
sbit SCL=P0^0;
```

```
////////////////////////////// Port #1 Definition Names ////////////////////////////////////
sbit LT1=P2^7;
sbit LT1FDB=P2^6;
sbit DR1=P2^5;
sbit DR1FDB=P2^4;
sbit LK1=P2^3;
sbit LK1FDB=P2^2;
sbit WD1=P2^1;
sbit WD1FDB=P2^0;
////////////////////////////// Port #2 Definition Names ////////////////////////////////////
sbit LT2=P3^0;
sbit LT2FDB=P3^1;
sbit DR2=P3^2;
sbit DR2FDB=P3^3;
sbit LK2=P3^4;
sbit LK2FDB=P3^5;
sbit WD2=P3^6;
sbit WD2FDB=P3^7;
////////////////////////////// Port #3 Definition Names ////////////////////////////////////
sbit LT3=P1^0;
sbit LT3FDB=P1^1;
sbit DR3=P1^4;
sbit DR3FDB=P1^5;
sbit LK3=P1^6;
sbit LK3FDB=P1^7;


////////////////////////////////////i2c buffer/////////////////////////////////////////////

void I2CBitDly()
 // wait approximately 4.7uS
{
// tune to xtal. This works at 11.0592MHz
```

```
unsigned int time_end = 10;
unsigned int index;
for (index = 0; index < time_end; index++);
return;
}


//////////////////////////////////////i2c START/////////////////////////////////////////////
void I2CSendStart()
{
  SDA = 1;            // i2c start bit sequence
  I2CBitDly();
  SCL = 1;
  I2CBitDly();
  SDA = 0;
  I2CBitDly();
  SCL = 0;
  I2CBitDly();
}
//////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////i2c Stop/////////////////////////////////////////////
void I2CSendStop()
{
  SDA = 0;            // i2c stop bit sequence
  I2CBitDly();
  SCL = 1;
  I2CBitDly();
  SDA = 1;
  I2CBitDly();
}
//////////////////////////////////////////////////////////////////////////////////////////////
```

```
//////////////////////Set SCL high, and wait for it to go high////////////////////////////////
void I2CSCLHigh(void)
{
register int err;
SCL = 1;
while (! SCL)
{
err++;
if (!err)
{
return;
}
}
}
////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////Send function/////////////////////////////////////////
void I2CSendByte(unsigned char bt)
{
register unsigned char i;
for (i=0; i<8; i++)
{
if (bt & 0x80) SDA = 1; // Send each bit, MSB first changed 0x80 to 0x01
else SDA = 0;
I2CSCLHigh();
I2CBitDly();
SCL = 0;
I2CBitDly();
bt = bt << 1;
}
SDA = 1; // Check for ACK
I2CBitDly();
I2CSCLHigh();
```

```
I2CBitDly();
if (SDA)
SCL = 0;
I2CBitDly();
SDA = 1; // end transmission
SCL = 1;
}
////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////// Transmit to SDA//////////////////////////////////////
void I2CSendAddr(unsigned char addr, unsigned char rd)
{
I2CSendStart();
I2CSendByte(addr+rd); // send address byte
}
////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////// Receive from SDA ////////////////////////////////////////
unsigned char I2CGetByte(unsigned char lastone) // last one == 1 for last byte; 0
for any other byte
{
register unsigned char i, res;
res = 0;
for (i=0;i<8;i++) // Each bit at a time, MSB first
{
I2CSCLHigh();
I2CBitDly();
res *= 2;
if (SDA) res++;
SCL = 0;
I2CBitDly();
}
SDA = lastone; // Send ACK according to 'lastone'
```

```
I2CSCLHigh();
I2CBitDly();
SCL = 0;
SDA = 1; // end transmission
SCL=1;
I2CBitDly();
return(res);
}
////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////SERIAL PORT/////////////////////////////////////////////
void Tx(unsigned char c)
{
   while (TI_0==0);
   TI_0=0;
   SBUF0=c;
}
////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////Digital Transfer/////////////////////////////////////////
void Tx_char(unsigned char c)
{
   unsigned char s[4];
   int i;

   c=c&0xff;
   sprintf(s,"%#x",(int)c);        // write into string.
   for (i=0;i<4;i++)
    Tx(s);
}
////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////String port/////////////////////////////////////////////
```

```
void Uart_Tx2(unsigned char x)
{
   while (TI_0==0);
   TI_0=0;
   SBUF0=x;
}
////////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////Send String/////////////////////////////////////////////////
void sendstring (char *String)
{
   int i=0;    //set counter to 0
   while(String)
   {
      Uart_Tx2(String[i++]);
   }

}
////////////////////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////Main Code/////////////////////////////////////////////////
void main ()
{
    unsigned char c;
    unsigned char addr;
    unsigned char lastone;
    unsigned char rd;

    SCON1 = 0x52;
    SCON0 = 0x52;
    TMOD  = 0x20;
    TCON  = 0x69;
    TH1   = 0xFD;
```

```
 delay(500);

 norm_show();
 printf("WELCOME!SYSTEM STARTUP...\n\r");
 delay(2000);
 lcd_clear();

 while(1)

 {

norm_show();

   c=getchar();

    if (c=='A')            //epilogi zwnis 1
   {
    c=getchar();

    if (c=='A')          //epilogi lights1
    {
     c=getchar();

     if (c=='0') LT1=0;
     else LT1=1;

     delay(500);

     if (LT1FDB==1)
     {
      lcd_clear();
      delay(1000);
      printf("\n\rERROR: LIGHTS1 NOT RESPONDING\n\r");
```

```
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLIGHTS1 OK!\n\r");
 }
}

if (c=='B')          //epilogi door1
{
 c=getchar();

 if (c=='0') DR1=0;
 else DR1=1;

 delay(500);

 if (DR1FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR1 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR1 OK!\n\r");
 }
}

if (c=='C')          //epilogi lock1
```

```
{
 c=getchar();

 if (c=='0') LK1=0;
 else LK1=1;

 delay(500);

 if (LK1FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: LOCK1 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLOCK1 OK!\n\r");
 }
}

if (c=='D')           //epilogi window1
{
 c=getchar();

 if (c=='0') WD1=0;
 else WD1=1;

 delay(500);

 if (WD1FDB==1)
 {
```

```
    lcd_clear();
    delay(1000);
    printf("\n\rERROR: WINDOW1 NOT RESPONDING\n\r");
   }
   else
   {
    lcd_clear();
    delay(1000);
    printf("\n\rWINDOW1 OK!\n\r");
   }
  }
}


else if(c=='B')         //epilogi zwnis2
{
  c=getchar();

  if (c=='A')          //epilogi lights2
  {
   c=getchar();

   if (c=='0') LT2=0;
   else LT2=1;

   delay(500);

   if (LT2FDB==1)
   {
    lcd_clear();
    delay(1000);
    printf("\n\rERROR: LIGHTS2 NOT RESPONDING\n\r");
   }
```

```
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLIGHTS2 OK!\n\r");
 }
}

if (c=='B')          //epilogi door2
{
 c=getchar();

 if (c=='0') DR2=0;
 else DR2=1;

 delay(500);

 if (DR2FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR2 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR2 OK!\n\r");
 }
}

if (c=='C')          //epilogi lock2
{
```

```
 c=getchar();

 if (c=='0') LK2=0;
 else LK2=1;

 delay(500);

 if (LK2FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: LOCK2 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLOCK2 OK!\n\r");
 }
}

if (c=='D')           //epilogi window2
{
 c=getchar();

 if (c=='0') WD2=0;
 else WD2=1;

 delay(500);

 if (WD2FDB==1)
 {
  lcd_clear();
```

```
    delay(1000);
    printf("\n\rERROR: WINDOW2 NOT RESPONDING\n\r");
   }
   else
   {
    lcd_clear();
    delay(1000);
    printf("\n\rWINDOW2 OK!\n\r");
   }
  }
}




else if(c=='C')         //epilogi zwnis3
{
  c=getchar();

  if (c=='A')          //epilogi lights3
  {
   c=getchar();

   if (c=='0') LT3=0;
   else LT3=1;

   delay(500);

   if (LT3FDB==1)
   {
    lcd_clear();
    delay(1000);
    printf("\n\rERROR: LIGHTS3 NOT RESPONDING\n\r");
```

```
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLIGHTS3 OK!\n\r");
 }
}

if (c=='B')          //epilogi door3
{
 c=getchar();

 if (c=='0') DR3=0;
 else DR3=1;

 delay(500);

 if (DR3FDB==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR3 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR3 OK!\n\r");
 }
}

if (c=='C')          //epilogi lock3
```

```
    {
     c=getchar();

     if (c=='0') LK3=0;
     else LK3=1;

     delay(500);

     if (LK3FDB==1)
     {
      lcd_clear();
      delay(1000);
      printf("\n\rERROR: LOCK3 NOT RESPONDING\n\r");
     }
     else
     {
      lcd_clear();
      delay(1000);
      printf("\n\rLOCK3 OK!\n\r");
     }
    }
  }
 if (c=='D')            //epilogi zwnis 4 (I2C)
  {
    c=getchar();

    if (c=='A')          //epilogi lights4
    {
     c=getchar();

     if (c=='0')
    {
          I2CBitDly();
```

```
      I2CSCLHigh();
      I2CSendStart();
      Tx_char(0);
      I2CSendAddr(addr,rd);
      I2CGetByte(lastone);
      I2CSendStop();
      delay(500);
     }
else
  {
      I2CBitDly();
      I2CSCLHigh();
      I2CSendStart();
      Tx_char(1);
      I2CSendAddr(addr,rd);
      I2CGetByte(lastone);
      I2CSendStop();
      delay(500);
      };

delay(500);

if (c==1)
{
 lcd_clear();
 delay(1000);
 printf("\n\rERROR: LIGHTS1 NOT RESPONDING\n\r");
}
else
{
 lcd_clear();
 delay(1000);
 printf("\n\rLIGHTS1 OK!\n\r");
```

```
 }
}

if (c=='B')          //epilogi door1
{
 c=getchar();

 if (c=='0')
{
      I2CBitDly();
      I2CSCLHigh();
      I2CSendStart();
      Tx_char(0);
      I2CSendAddr(addr,rd);
      I2CGetByte(lastone);
      I2CSendStop();
      delay(500);
     }
 else
   {
      I2CBitDly();
      I2CSCLHigh();
      I2CSendStart();
      Tx_char(1);
      I2CSendAddr(addr,rd);
      I2CGetByte(lastone);
      I2CSendStop();
      delay(500);
      };

 delay(500);

 if (c==1)
```

```
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: DOOR1 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rDOOR1 OK!\n\r");
 }
}

if (c=='0')
{
      I2CBitDly();
      I2CSCLHigh();
      I2CSendStart();
      Tx_char(0);
      I2CSendAddr(addr,rd);
      I2CGetByte(lastone);
      I2CSendStop();
      delay(500);
     }
 else
   {
      I2CBitDly();
      I2CSCLHigh();
      I2CSendStart();
      Tx_char(1);
      I2CSendAddr(addr,rd);
      I2CGetByte(lastone);
      I2CSendStop();
```

```
        delay(500);
        };

 delay(500);

 if (c==1)
 {
  lcd_clear();
  delay(1000);
  printf("\n\rERROR: LOCK1 NOT RESPONDING\n\r");
 }
 else
 {
  lcd_clear();
  delay(1000);
  printf("\n\rLOCK1 OK!\n\r");
 }
}

if (c=='D')           //epilogi window1
{
 c=getchar();

if (c=='0')
{
        I2CBitDly();
        I2CSCLHigh();
        I2CSendStart();
        Tx_char(0);
        I2CSendAddr(addr,rd);
        I2CGetByte(lastone);
        I2CSendStop();
        delay(500);
```

```
        }
      else
        {
          I2CBitDly();
          I2CSCLHigh();
          I2CSendStart();
          Tx_char(1);
          I2CSendAddr(addr,rd);
          I2CGetByte(lastone);
          I2CSendStop();
          delay(500);
          };

      delay(500);

      if (c==1)
      {
       lcd_clear();
       delay(1000);
       printf("\n\rERROR: WINDOW1 NOT RESPONDING\n\r");
      }
      else
      {
       lcd_clear();
       delay(1000);
       printf("\n\rWINDOW1 OK!\n\r");
      }
     }
    }



}
```