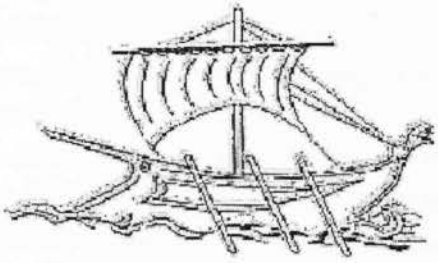
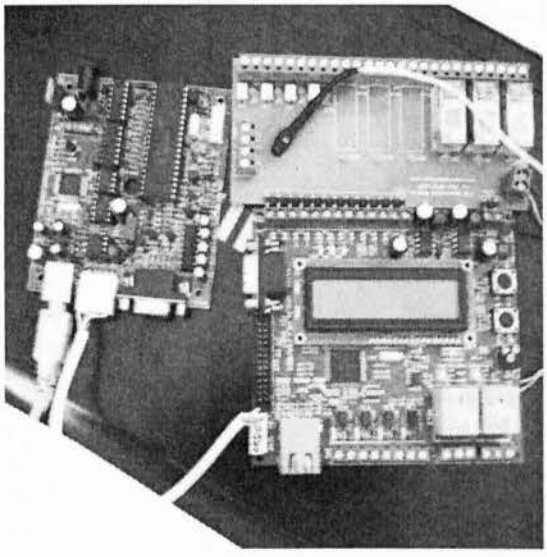
ΑΤΕΙ ΠΕΙΡΑΙΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Διάταξη ελέγχου και απομακρυσμένης διαχείρισης μέσω διαδικτύου αυτοματισμών εγκατάστασης θερμοκηπίου»



Γαλιατσάτου Ανδριανή
ΑΜ: 34420

Επιβλέπων καθηγητής
Τσελές Δημήτριος

Αθήνα, Οκτώβριος 2012

Συνοπτική περίληψη

Η πτυχιακή αυτή εργασία έχει ως αντικείμενο τον προγραμματισμό διάταξης βασισμένης σε μικροεπεξεργαστή PIC18F63J90 αρχιτεκτονικής 8-bit της Microchip. Η διάταξη που κατασκευάζεται από την εταιρεία OLIMEX σε ιδιαίτερα χαμηλό κόστος είναι εφοδιασμένη με εξόδους ρελαί και ηλεκτρικά απομονωμένες εισόδους, κουμπιά δοκιμών, ποτενσιόμετρο για ανάπτυξη εφαρμογών με αναλογικά μεγέθη και ενσωματωμένο αισθητήρα θερμοκρασίας για μέτρηση της θερμοκρασίας περιβάλλοντος. Με βάση το έτοιμο λογισμικό και βιβλιοθήκες σε γλώσσα C και το περιβάλλον προγραμματισμού MPLAB-IDE που δίνει δωρεάν η Microchip στην ιστοσελίδα της για πλήρη ανάπτυξη των βασικότερων πρωτοκόλλων του TCP/IP (HTTP, SMTP, FTP, TELNET κλπ), στα πλαίσια της πτυχιακής θα υλοποιηθεί εφαρμογή αυτοματισμού του θερμοκηπίου για έλεγχο και απομακρυσμένη επίβλεψη κάποιων ενδεικτικών λειτουργιών ποτίσματος και ελέγχου θερμοκρασίας, ενώ θα υλοποιηθεί και ιστοσελίδα σε γλώσσα HTML για τον ενσωματωμένο στην πλακέτα WEB server μέσω της οποίας θα γίνεται απομακρυσμένη επίβλεψη και τηλεέλεγχος της εγκατάστασης από κάποιο υπολογιστή ή κινητό τηλέφωνο με τη χρήση οποιουδήποτε κοινού προγράμματος περιήγησης (για παράδειγμα Internet Explorer, Mozilla Firefox, Opera κλπ.)

Περιεχόμενα

Εισαγωγή

Σε αυτή την πτυχιακή θα δούμε πώς είναι δυνατό μια αγροτική καλλιέργεια να την ελέγχουμε από απόσταση μέσω διαδικτύου. Για να το πετύχουμε αυτό θα χρησιμοποιήσουμε την πλακέτα PIC-MAXI-WEB της OLIMEX και θα προγραμματίσουμε τον μικροεπεξεργαστή της όπως απαιτείται για την καλύτερη και σωστή λειτουργία της, επίσης θα χρησιμοποιήσουμε μια δεξαμενή για το νερό, δύο floater ένα στο κάτω και ένα στο πάνω μέρος της δεξαμενής για τον έλεγχο της στάθμης του νερού, μια αντλία για την άντληση του νερού από τη δεξαμενή, μια αντλία για το πότισμα, έναν ανεμιστήρα και ένα boiler για την ρύθμιση της θερμοκρασίας.

Η συγκεκριμένη ,υποθετική, αγροτική καλλιέργεια είναι μικρή και περιλαμβάνει ένα θερμοκήπιο εντός του οποίου γίνεται το πότισμα των φυτών και η διατήρηση της θερμοκρασίας του στα επιθυμητά όρια για την καλύτερη ανάπτυξη των φυτών .

Στόχος είναι ο έλεγχος της θερμοκρασίας του θερμοκηπίου, εντός των επιθυμητών τιμών, καθώς και το κάθε πότε θα γίνεται το πότισμα των φυτών.

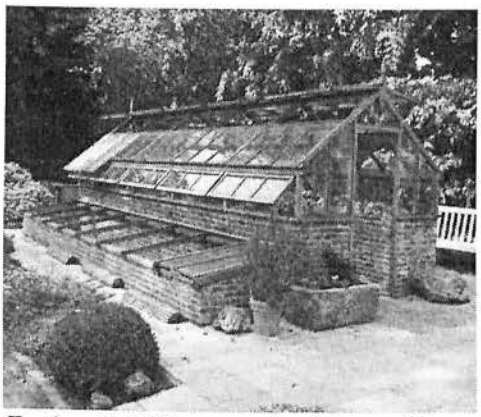
Για να το πετύχουμε αυτό αρχικά θα πρέπει να μελετήσουμε τον τρόπο λειτουργίας ενός θερμοκηπίου, ενός αυτόματου ποτίσματος και τη διαδικασία άντλησης νερού από μια δεξαμενή. Στη συνέχεια, σύμφωνα με αυτά που θα μελετήσαμε θα πρέπει να δούμε πώς θα υλοποιήσουμε την εφαρμογή μας και τη υλικά θα χρησιμοποιήσουμε.

Κεφάλαιο 1

Θερμοκήπια και συστήματα αυτόματου ποτίσματος

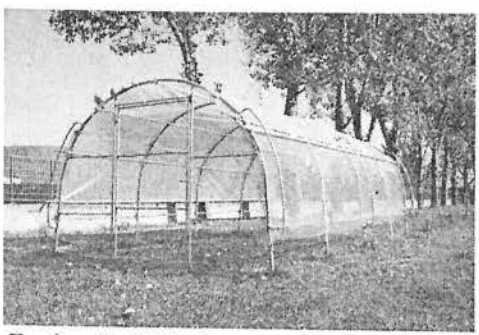
1.1. Η λειτουργία του Θερμοκηπίου

Τα θερμοκήπια είναι κλειστές, διαφανείς κατασκευές που ο σκελετός τους αποτελείται από σιδερένια ή ξύλινα δοκάρια, καλύπτεται από πλαστικά φύλλα ή γυαλί που έχουν σαν σκοπό τη δημιουργία όσο το δυνατόν πιο ευνοϊκών συνθηκών περιβάλλοντος για την καλλιέργεια των φυτών, ιδιαίτερα σε εποχές που στην ύπαιθρο αυτό δεν συμβαίνει και σε περιοχές που δεν είναι ενδεδειγμένες για την καλλιέργεια κάποιου είδους στις επικρατούσες φυσικές οικολογικές συνθήκες. Για το λόγο αυτό τα θερμοκήπια των βόρειων χωρών έχουν βαριές κατασκευές και, πολλές φορές αποτελούνται από διπλά τζάμια και διπλή οροφή. Τα θερμοκήπια αυτά θερμαίνονται. Αντίθετα, στις νότιες περιοχές της Ελλάδος, όπως για παράδειγμα στη νότια Μεσσηνία και στην Κρήτη, οι κατασκευές είναι πολύ ελαφριές, αποτελούνται από πλαστικό απλωμένο πάνω σε ξύλινο σκελετό, χωρίς να θερμαίνεται.



Εικόνα 1: Ξύλινος σκελετός με γυάλινα φύλλα

Εικόνα 2: Μεταλλικός σκελετός με γυάλινα φύλλα



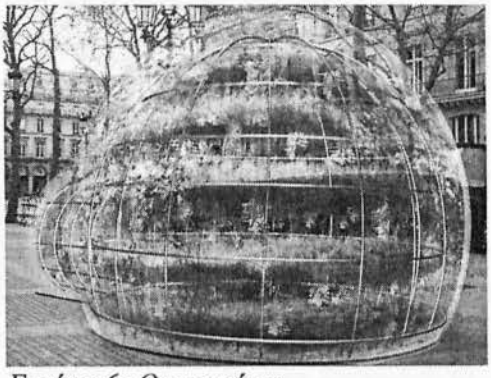
Εικόνα 3: Σιδερένιος σκελετός με πλαστικά φύλλα

Εικόνα 4: Μεγάλη εγκατάσταση θερμοκηπίων

Το θερμοκήπιο αποσκοπεί, δηλαδή, στην δημιουργία κατάλληλου μικροκλίματος για την ανάπτυξη των φυτών, όταν οι καιρικές συνθήκες δεν επιτρέπουν την καλλιέργεια αυτών στην ύπαιθρο. Υπενθυμίζεται ότι ως μικροκλίμα ορίζεται το σύνολο των κλιματολογικών συνθηκών που επικρατούν σε ένα ομοιογενή χώρο περιορισμένης έκτασης κοντά στην επιφάνεια του εδάφους.

Η αρχή λειτουργίας των θερμοκηπίων είναι η εξής: το ηλιακό φως ("ορατή ακτινοβολία") περνά μέσα από το τζάμι του θερμοκηπίου και απορροφάται από τα φυτά που βρίσκονται μέσα. Τα φυτά παράγουν θερμότητα ("υπέρυθρες ακτινοβολίες"), αυτή είναι αόρατη, αλλά μπορούμε να την αισθανθούμε. Μέρος της θερμότητας επιστρέφει έξω από το θερμοκήπιο, αλλά όχι όλη. Το τζάμι του θερμοκηπίου αντανακλά μερικές από τις υπέρυθρες ακτινοβολίες πίσω, έτσι το εσωτερικό του θερμοκηπίου θερμαίνεται.

Εικόνα 5: Θερμοκήπιο

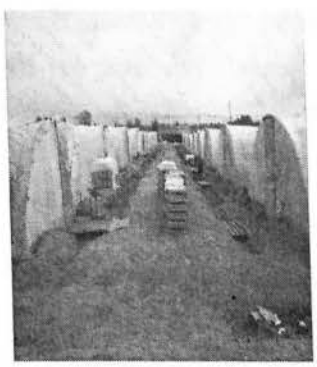


Εικόνα 6: Θερμοκήπιο

Στα θερμοκήπια ειδικότερα, οι κλιματικοί παράγοντες που επηρεάζονται και τροποποιούνται σύμφωνα με τις απαιτήσεις των φυτών είναι κυρίως η θερμοκρασία (μέσω θέρμανσης, εξαερισμού, σκίασης, κλπ.) και δευτερευόντως η ατμοσφαιρική υγρασία και η διάρκεια του φωτισμού (φωτοπερίοδος). Παράλληλα, μέσα στο χώρο των θερμοκηπίων καθίσταται επιπλέον δυνατή και η τροποποίηση της συγκέντρωσης διοξειδίου του άνθρακα καθώς και η ένταση του φωτισμού. Η ρύθμιση αυτών των εσωτερικών συνθηκών λειτουργίας του θερμοκηπίου απαιτεί τον κατάλληλο εξοπλισμό.

Στον απαραίτητο εξοπλισμό ενός σύγχρονου θερμοκηπίου περιλαμβάνονται το σύστημα θέρμανσης, το σύστημα αερισμού και ένα σύστημα ρύθμισης της σχετικής υγρασίας του χώρου. Σε ορισμένες περιπτώσεις μπορεί να απαιτηθεί και ένα σύστημα παροχής τεχνητού φωτισμού καθώς και εγκαταστάσεις εμπλουτισμού του αέρα με CO2. Στον απαραίτητο εξοπλισμό των θερμοκηπίων συμπεριλαμβάνονται επίσης και τα συστήματα άρδευσης και υδρολίπανσης των φυτών.

Ωστόσο όπως φαίνεται και στις παρακάτω φωτογραφίες ένα θερμοκήπιο μπορεί να είναι πολύ μικρό ή και πολύ μεγάλο:


Η ιστορία των θερμοκηπίων δεν είναι υπόθεση των τελευταίων χρόνων. Ένα από τα πρώτα θερμοκήπια που φτιάχτηκαν στην Ευρώπη ήταν στη Βοημία περίπου το 1680 . Στο θερμοκήπιο αυτό καλλιεργήθηκαν οι πρώτες ορχιδέες στην Ευρώπη. Αργότερα, περί το 1750, ο πρίγκιπας του Λιχτενστάιν έφτιαξε το πρώτο μεγάλο και θερμαινόμενο θερμοκήπιο στην Ευρώπη στην πόλη Λέντιτσε (*Lednice*) στη νότια Τσεχία.

1.2. Αυτόματο σύστημα ποτίσματος

Το αυτόματο πότισμα είναι ένα είδος συστήματος άρδευσης το οποίο λειτουργεί με τη βοήθεια ενός ηλεκτρονικού ελεγκτή. Αυτός ο τύπος συστήματος είναι ιδανικός για όσους ταξιδεύουν συχνά και δε μπορούν να ποτίσουν τα φυτά τους σε τακτική βάση ,είναι επίσης πολύ βολικό και διευκολύνει εκείνους που έχουν γεωργικές καλλιέργειες, αγροκτήματα και γενικά μεγάλους χώρους για καλλιέργεια.

Όταν εγκατασταθεί ένα αυτόματο σύστημα άρδευσης μπορεί να προγραμματιστεί για να ποτίζει ανά συγκεκριμένα χρονικά διαστήματα ,μέσω του συστήματος αυτού εξοικονομείται νερό. Καθώς υπάρχει δυνατότητα ρύθμισης της ποσότητας νερού ποτίσματος ανάλογα με το είδος της καλλιέργειας. Η συγκεκριμένη λειτουργία, το καθιστά μια οικονομική και περιβαλλοντική επιλογή.



Οι ιδιοκτήτες Γής πριν εγκαταστήσουν ένα αυτόματο σύστημα ποτίσματος πρέπει να χαρτογραφήσουν την καλλιέργεια τους, αυτό θα τους βοηθήσει να καθορίσουν την ποσότητα του νερού σε κάθε τμήμα της καλλιέργειας τους. Το καλύτερο είναι να κατηγοριοποιήσουν τα φυτά ανάλογα την αντοχή τους στην ξηρασία και ανάλογα με το ποιά

χρειάζονται περισσότερο νερό. Τα φυτά που έχουν παρόμοιες ανάγκες θα πρέπει να φυτεύονται μαζί προκειμένου να επιτευχθεί η βέλτιστη υγεία των φυτών.

Μετά τη χαρτογράφηση της καλλιέργειας το επόμενο βήμα είναι να αποφασιστεί ο τύπος του αυτόματου συστήματος άρδευσης που πρέπει να εγκατασταθεί. Τα συστήματα ποτίσματος συνήθως είναι δύο ειδών, α) άρδευσης με τεχνητή βροχή («sprinkler irrigation») και β) στάγδην άρδευσης («drip irrigation»). Ο κάθε τύπος έχει πολλά διαφορετικά γνωρίσματα και οφέλη.

Στην άρδευση με **τεχνητή βροχή**, το νερό εφαρμόζεται στον υπό άρδευση χώρο με μορφή τεχνητής βροχής. Πρόκειται για συστήματα που απαιτούν για τη λειτουργία τους υψηλή πίεση και παροχή.

Σε γενικές γραμμές, τα συστήματα αυτά μπορούν να χωριστούν ,ανάλογα με το εάν παραμένουν στον υπό άρδευση χώρο οι γραμμές και το σύστημα άρδευσης και διακρίνονται σε μόνιμα, ημιμόνιμα και κινητά/αυτοκινούμενα.

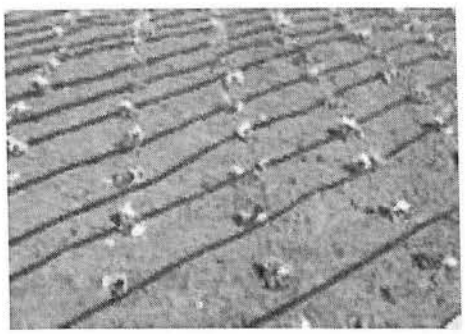
Εικόνα 7: Τεχνητή βροχή ("sprinkler irrigation")

Εικόνα 8: Τεχνητή βροχή ("sprinkler irrigation")

Η **στάγδην άρδευση** μπορεί να είναι μια από τις καλύτερες μεθόδους για πότισμα γκαζόν, κήπων και παρτεριών λουλουδιών. Ένα καλά σχεδιασμένο αυτόματο σύστημα στάγδην άρδευσης εξασφαλίζει ελάχιστη απώλεια νερού εξαιτίας της εξάτμισης, επιπλέον εξασφαλίζεται ότι το μεγαλύτερο μέρος του νερού χρησιμοποιείται αποτελεσματικά και ποτίζει ακριβώς τα φυτά που χρειάζεται χωρίς να πηγαίνει χαμένο.

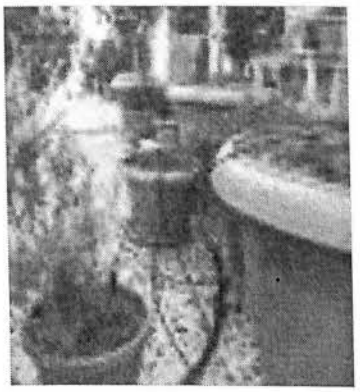
Όσον αφορά την χρήση του σε σπίτια είναι συνήθως καλύτερο το αυτόματο σύστημα να θαφτεί κάτω από το έδαφος, αυτό το προστατεύει στο να μη καταστραφεί από ζώα και από οποιονδήποτε μπορεί να περπατήσει στο χώρο που έχει τοποθετηθεί.

Το σύστημα αυτό είναι ευεργετικό γιατί εκθέτει τις ρίζες του άμεσα στο νερό αντί να περιμένει να εισχώρηση το νερό από την επιφάνεια του εδάφους στο εσωτερικό.
Αυτή η μέθοδος είναι ιδιαίτερα ωφέλιμη κατά τη διάρκεια ξηρών καιρικών συνθηκών.



Εικόνα 9: Στάγδην άρδρευσης ("drip irrigation")

Εικόνα 10: Στάγδην άρδρευσης ("drip irrigation")



Κεφάλαιο 2

Πώς θα λειτουργεί η αγροτική καλλιέργεια της συγκεκριμένης εφαρμογής

Για τη συγκεκριμένη εφαρμογή υποθέτουμε ότι η παρούσα αγροτική καλλιέργεια είναι αρκετά μικρή και οι συνθήκες όσον αφορά την ατμοσφαιρική υγρασία, τη διάρκεια και την ένταση του φωτισμού καθώς και τη συγκέντρωση διοξειδίου του άνθρακα στην ατμόσφαιρα είναι οριακά ιδανικές. Το πότισμα, για τις απαιτήσεις του εδάφους , θα πραγματοποιείται για πέντε λεπτά από δύο φορές την μέρα. Το μόνο που θα χρειαστεί να ελέγχουμε είναι η θερμοκρασία του θερμοκηπίου.

Έτσι λοιπόν για την υλοποίηση της εφαρμογής θα χρειαστούμε μια δεξαμενή με νερό για το πότισμα, δύο floater εντός της δεξαμενής, ένα στο κάτω μέρος της δεξαμενής για τον έλεγχο αν είναι άδεια η δεξαμενή έτσι ώστε αν είναι άδεια να ξεκινάει το γέμισμα της και να σταματάει το πότισμα αν πραγματοποιείται εκείνη τη στιγμή προκειμένου να μη καεί η αντλία ποτίσματος αφού θα είναι σε λειτουργία και δε θα έχει νερό η δεξαμενή για να τραβήξει και ένα στο πάνω μέρος της δεξαμενής για το αν είναι γεμάτη, έτσι ώστε να σταματάει το γέμισμά προκειμένου να αποφευχθεί η υπερχείλιση. Επίσης μια αντλία για το γέμισμα της δεξαμενής με νερό που θα γίνεται αυτόματα κάθε φορά που θα έχει αδειάσει η δεξαμενή, μια αντλία για την άντληση του νερού από τη δεξαμενή για το πότισμα, ένας ανεμιστήρας για πτώση της θερμοκρασίας και ένα boiler για την αύξηση της θερμοκρασίας εντός του θερμοκηπίου σε περίπτωση που η θερμοκρασία δεν είναι εντός του προβλεπόμενου ορίου, το οποίο, έχουμε τη δυνατότητα να του ορίζουμε κάθε φορά τη τιμή που θέλουμε εμείς μέσω ενός ποτενσιόμετρου που βρίσκεται στην πλακέτα. Ο έλεγχος της θερμοκρασίας γίνεται μέσω αισθητηρίου που βρίσκεται ενσωματωμένο επίσης πάνω στη πλακέτα. Τέλος τα υλικά που είναι απαραίτητα για την υλοποίηση του αυτοματισμού της συγκεκριμένης εφαρμογής, αναφέρονται αναλυτικά στο κεφάλαιο 3.

Αυτό που θα χρειαστεί να τροποποιήσουμε για τις ανάγκες της παρουσίασης είναι οι χρόνοι ποτίσματος για λόγους διευκόλυνσης μας, οι 2 είσοδοι του συστήματος μας που αντιστοιχούν στα δύο floater και θα είναι τα buttons της πλακέτας , δηλαδή στο floater1 θα αντιστοιχεί το BUTTON0 και στο floater2 το BUTTON1 αντίστοιχα, και το αποτέλεσμα των εξόδων θα είναι εμφανές από την ενεργοποίηση και απενεργοποίηση του αντίστοιχου rele, το ίδιο ισχύει για το πότισμα, τον ανεμιστήρα και το boiler .

Έτσι λοιπόν προγραμματίζοντας τη πλακέτα και ορίζοντας μέσω του ποτενσιόμετρου ως επιθυμητή θερμοκρασία θερμοκηπίου έστω τους 28 βαθμούς Κελσίου θέτουμε την εφαρμογή σε λειτουργία.
Όσον αφορά τον έλεγχο της θερμοκρασίας του θερμοκηπίου γίνεται ως εξής: εμείς έχουμε ορίσει όπως προαναφέραμε επιθυμητή θερμοκρασία τους 28 βαθμούς, μέσω του ενσωματωμένου αισθητηρίου-θερμόμετρο στην πλακέτα ελέγχεται συνεχώς η πραγματική θερμοκρασία στο χώρο, αν αυτή υπερβεί τους 28 βαθμούς τότε αυτόματα ενεργοποιείται το

σύστημα ψύξης δηλαδή ο ανεμιστήρας. Ταυτόχρονα εξακολουθείται να ελέγχεται η θερμοκρασία μέσω του θερμομέτρου ,όταν αυτή επανέλθει στους 28 βαθμούς ο ανεμιστήρας απενεργοποιείται. Ενώ αν η θερμοκρασία του θερμοκηπίου πέσει κάτω από τους 28 βαθμούς τότε ενεργοποιείται αντίστοιχα το σύστημα θέρμανσης δηλαδή το boiler μέχρις ότου η θερμοκρασία να επανέλθει στην επιθυμητή τιμή δηλαδή στους 28 βαθμούς όπου και απενεργοποιείται το boiler.

Παράλληλα με τη διαδικασία ρύθμισης της θερμοκρασίας τρέχει και η διαδικασία του αυτόματου ποτίσματος.
Έστω ότι η δεξαμενή είναι ήδη γεμάτη, μέχρι το floater που βρίσκεται στο πάνω μέρος της δεξαμενής έτσι αρχίζει να ποτίζει το σύστημα την καλλιέργεια μας δύο φορές τη μέρα, κάθε δώδεκα ώρες για πέντε λεπτά κάθε φορά.
Κάποια στιγμή καθώς πραγματοποιούνται τα ποτίσματα η στάθμη της δεξαμενής θα κατέβει κάτω από το πάνω floater και αργότερα θα κατέβει κάτω και από το δεύτερο , τη στιγμή που θα πέσει κάτω από το δεύτερο floater αν εξελίσσεται η διαδικασία του ποτίσματος σταματάει προκειμένου να μην καεί η αντλία ποτίσματος καθώς θα λειτουργεί χωρίς νερό, επιπλέον ενεργοποιείται η διαδικασία γεμίσματος της δεξαμενής με νερό μέχρις ότου να φτάσει η στάθμη στο πάνω floater όπου και σταματάει η διαδικασία του γεμίσματος της δεξαμενής.

Όλες οι παραπάνω διαδικασίες και όλοι οι έλεγχοι τρέχουν συνεχώς μέσα στο πρόγραμμα με αποτέλεσμα να λειτουργεί σωστά το θερμοκήπιο μας και να μπορούν να αναπτυχθούν καλά τα φυτά μας.

Κεφάλαιο3

Υλοποίηση συστήματος αυτόματου ποτίσματος και θερμοκηπίου

3.1. Διαδικασία υλοποίησης - Βήματα

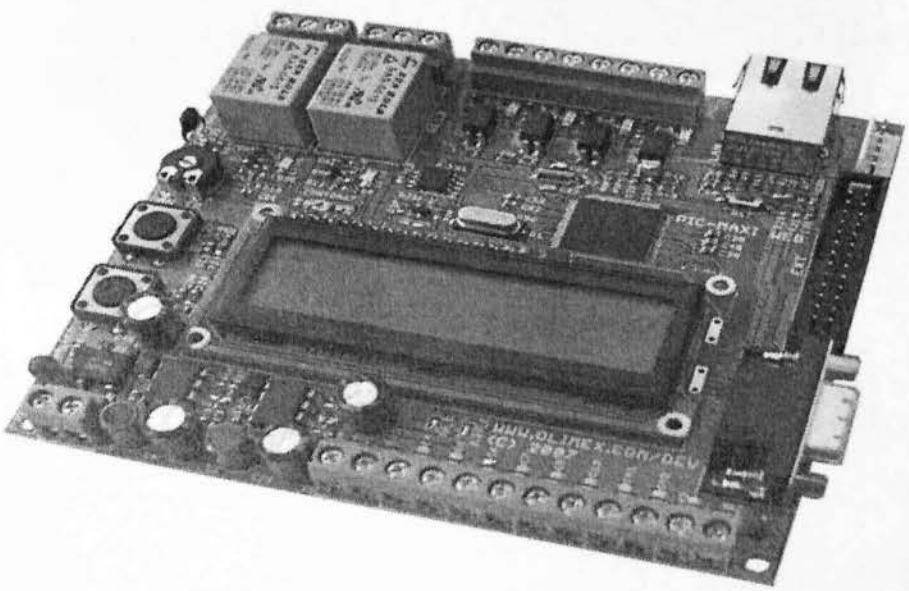
Για τη διαδικασία της υλοποίησης της εφαρμογής ακολουθήθηκαν τα εξής βήματα:

Βήμα 1: Σύνδεση της πλακέτας με την I/O Expansion Board με την πλακέτα PIC-MAXI-WEB
Βήμα 2: Σύνδεση της πλακέτας με τον προγραμματιστή, το τροφοδοτικό και κατ' επέκταση με τον υπολογιστή.
Βήμα 3:Έλεγχος σωστής λειτουργίας όλων των εισόδων και των εξόδων των πλακετών.
Βήμα 4:Προγραμματισμός πλακέτας για τη λειτουργία του αυτόματου ποτίσματος.
Βήμα 5:Προγραμματισμός πλακέτας για τη λειτουργία του θερμοκηπίου.
Βήμα 6:Δημιουργία ιστοσελίδας.

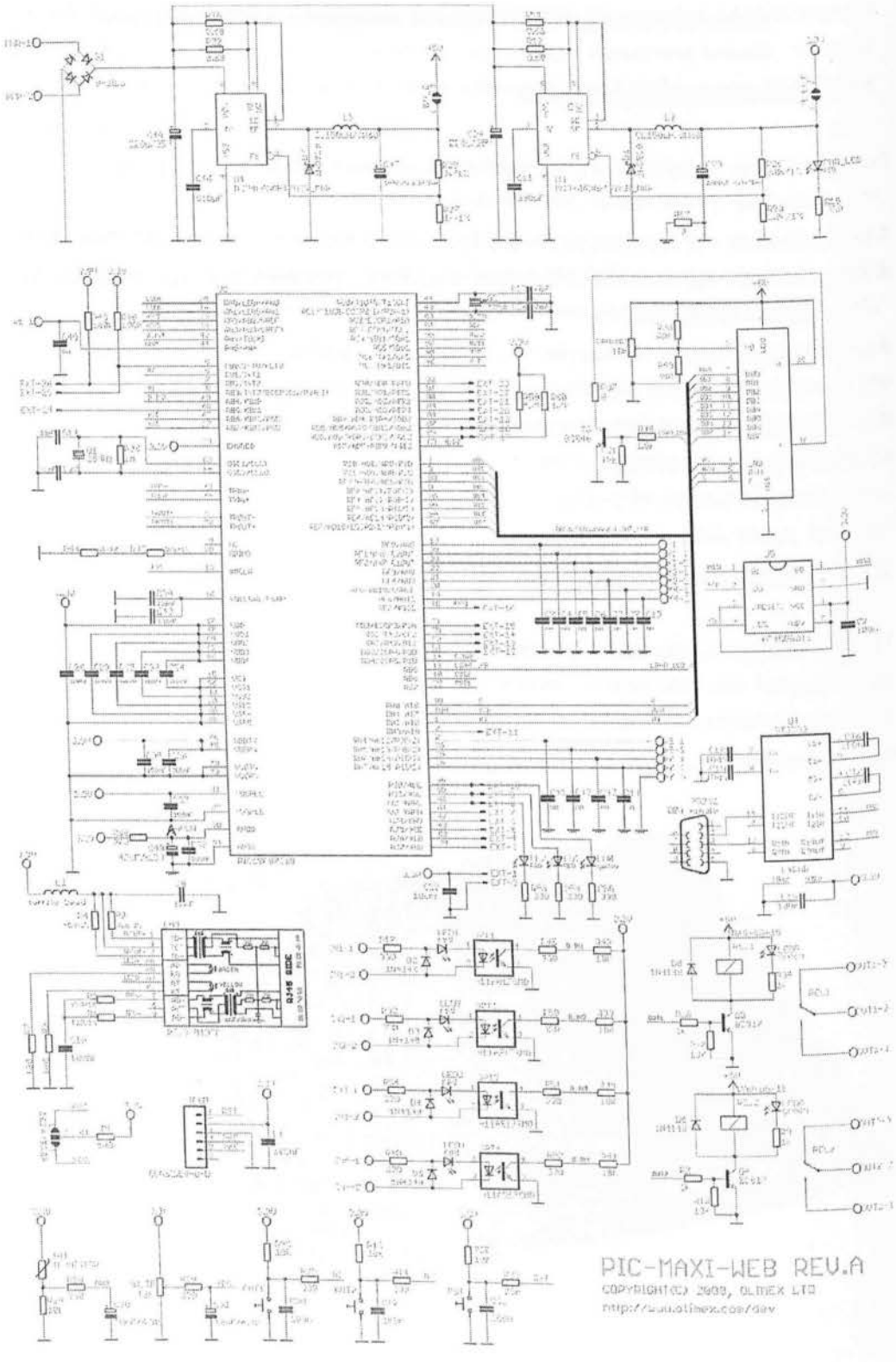
3.2. Υλικά που χρησιμοποιήθηκαν

Τα υλικά που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής είναι τα έξης:

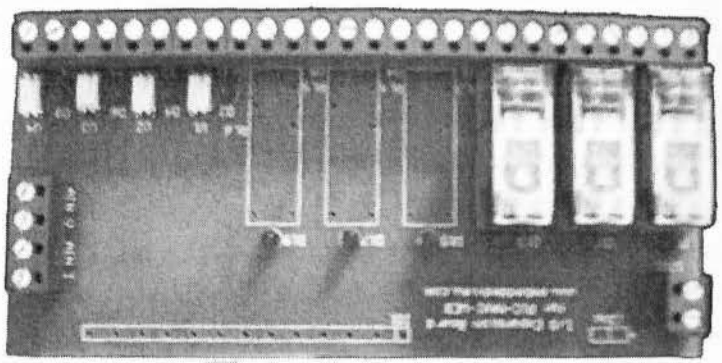
Μια πλακέτα **PIC-MAXI-WEB**:



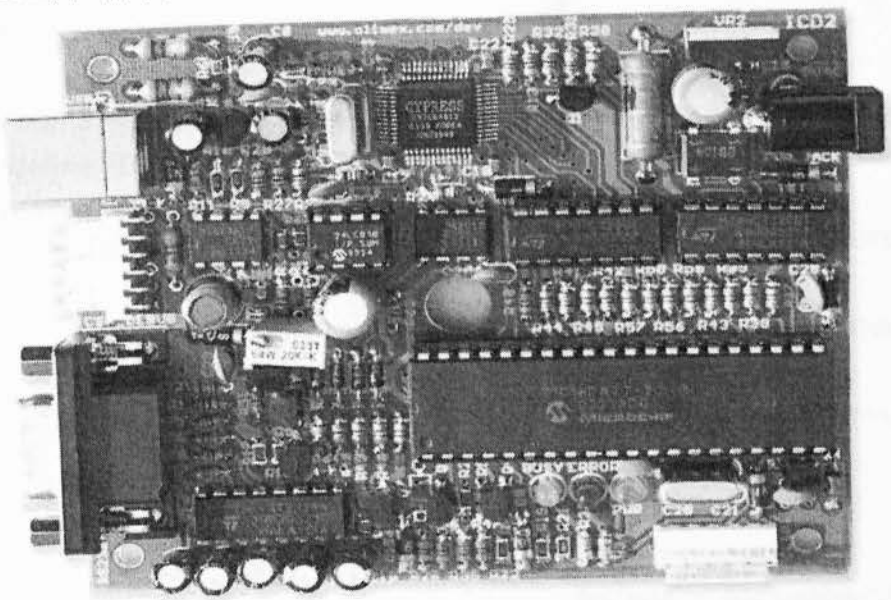
Κύκλωμα πλακέτας:



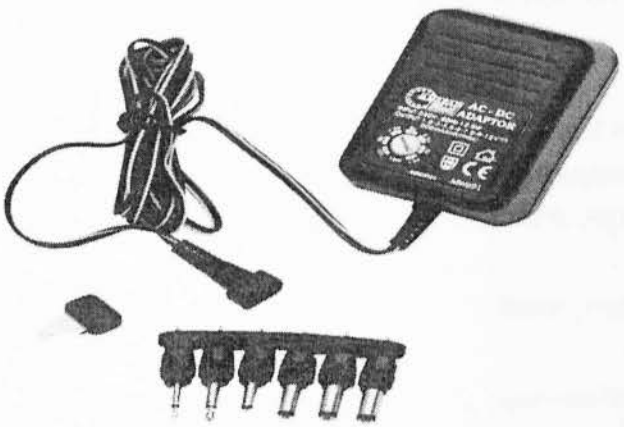
I/O Expansion Board:



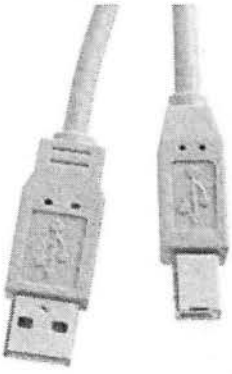
Ένας προγραμματιστής **PIC ICD2**:



Ένα τροφοδοτικό 12volt:



Ενα καλώδιο USB για επικοινωνία της πλακέτας με τον υπολογιστή:



Ενα καλώδιο Ethernet:

Κεφάλαιο 4

Δημιουργία ανάλυση κώδικα λειτουργίας, για το θερμοκήπιο και το αυτόματο πότισμα

Ο κώδικας που χρησιμοποιήθηκε για τη δημιουργία της εφαρμογής είναι γραμμένος σε γλώσσα C. Ένα κομμάτι του κώδικα χρησιμοποιήθηκε έτοιμο, κάποιο αλλάχτηκε και κάποιο γράφτηκε εξολοκλήρου από την αρχή. Συνθέτοντας τα στοιχεία αυτά βγήκε ο τελικός επιθυμητός κώδικας. Με μπλε χρώμα είναι τα στοιχεία που τροποποίησα ή έγραψα από την αρχή.Ολόκληρος ο κώδικας βρίσκεται στο παράρτημα Α στο τέλος στην σελίδα .Ακολουθεί η ανάλυση τών κομματιών του κώδικα που χρησιμοποιήθηκαν για τη συγκεκριμένη εφαρμογή .

Ανάλυση κώδικα

```
#define PUMP_RELAY INREL1_IO
#define POTISMA_RELAY INREL2_IO
#define ANEMISTHRAS_RELAY EXTREL1_IO
#define BOILER_RELAY EXTREL2_IO
```

Ορίζουμε τις τέσσερις εξόδους ως:
-αντλία γεμίσματος =PUMP_RELAY το INREL1_IO
-αντλία ποτίσματος=POTISMA_RELAY το INRELAY2_IO
-ανεμηστήρα=ANEMITHRAS_RELAY το EXTERNAL1_IO
-boiler=BOILER_RELAY το EXTERNAL2_IO

```
#define FLOTER1 BUTTON0 _IO
#define FLOTER2 BUTTON1_IO
```

Ορίζουμε τις δύο εισόδους.-floater ως:
-πάνω floater=FLOATER1 το BUTTON0_IO
-κάτω floater=FLOATER2 το BUTTON1_IO
Τα BUTTON έχουν ανάστροφη λογική

```
BYTE TIMER_POTISMA_ENABLE;

int TIMER_POTISMA;

int dec;
int ypol;
int potensio;
```

Ορίζουμε τις έξι μεταβλητές που θα χρησιμοποιήσουμε:
TIMER_POTISMA_ENABLE=μεταβλητή που πέρνει τιμές 0 και 1
TIMER_POTISMA =μετρητής,μετράει χρόνο
dec=πηλίκο
ypol=υπόλοιπο
potensio=η τιμή του ποτενσιομέτρου

```
int thermokrasia;
```
thermokrasia=η τιμή της θερμοκρασίας

```
strcpypgm2ram((char*)LCDText, "TEI PEIRAIA    "
                                        " A.Galiatsatou ");
```

Εμφανίζεται στην οθόνη ότι περιέχεται μέσα στα " "

```
INREL1_IO=0;
INREL2_IO=0;
EXTREL1_IO=0;
EXTREL2_IO=0;
EXTREL3_IO=0;
EXTREL4_IO=0;
EXTREL5_IO=0;
EXTREL6_IO=0;
```
Αρχικοποιούμε τις τιμές των εισόδων και των εξόδων.

```
while(1)
  {
if(TickGet() - t >= TICK_SECOND)
 {
  t = TickGet();
  LED0_IO ^= 1;
```
Με αυτό το if, το πρόγραμμα εντός του if, επαναλαμβάνεται κάθε ενα δευτερόλεπτο και όχι συνέχεια.

```
///********************  ARXH PTYXIAKHS  **********************///

//ANDR****************EKTELEITAI KATHE DEFTEROLEPTO************//

if (TIMER_POTISMA_ENABLE)TIMER_POTISMA++;
if (!TIMER_POTISMA_ENABLE)TIMER_POTISMA=0;

dec=TIMER_POTISMA/10;
ypol=TIMER_POTISMA%10;

/********dokimes********/
/*
if(potensio>450)
EXTREL3_IO=1;
```
Δοκιμές με μικρά προγραμματάκια για τον

```
else
EXTREL3_IO=0;
*/
/*
 if (thermokrasia>potensio)
    {EXTREL2_IO=1;}
  if (thermokrasia<=potensio)
    {EXTREL2_IO=0;}
*/
/**********************/
```

έλεγχο αν λειτουργούν σωστά το ποτενσιόμετρο και το θερμόμετρο.

```
/**********************THERMOKHPIO**************************/
if    (thermokrasia>((potensio/10)+1))
{ANEMISTHRAS_RELAY=1;}
 if (thermokrasia<=(potensio/10))
    {ANEMISTHRAS_RELAY=0;}
 if (thermokrasia<((potensio/10)-10))
    {BOILER_RELAY=1;}
 if (thermokrasia>=(potensio/10))
    {BOILER_RELAY=0;}
```

Ορίζουμε τη θερμοκρασία του potensio σε όποια τιμή επιθυμούμε και κατόπιν τη συγκρίνουμε με τη τιμή thermokrasia η οποία είναι και η πραγματική θερμοκρασία του χώρου. Το potensio είναι το ποτενσιόμετρο το οποίο παίρνει τιμές από 0-1024 και έχει συνεχώς μια μεταβολή γύρω στις 8-10 μονάδες, γι' αυτό για διευκόλυνση μας αλλά και για μεγαλύτερη ακρίβεια διαιρούμε νε το 10 καιθεωρούμε έτσι με υποδιαστολή στο τελευταίο ψηφίο, δηλαδή Το 245=24,5 , 32=3,2

```
/*************************************************************/

///**********************AYTOMATO POTISMA*********************///

if (FLOTER1 && FLOTER2)
{
    PUMP_RELAY=1;
     strcpypgm2ram((char*)LCDText, "DEKSAMENH      "
                                   "GEMIZEI      ");
```

Άν δεν έχω floater1 και δεν έχω floater2.

Ενεργοποιήται η αντλία για γέμισμα δεξαμενής.

Με αυτές τις δύο εντολές γράφουμε στην οθόνη στις θέσεις 28 και 29 το (dec+48) και το (ypol+48) αντίστοιχα, χωρίς το +48 θα μας εμφάνιζε τον κώδικα ASC||.

```
LCDText[28]=(48+dec);
LCDText[29]=(48+ypol);
```

```
    LCDUpdate();
   POTISMA_RELAY=0;              Απενεργοποιείται το πότισμα.
   TIMER_POTISMA_ENABLE=0;       Απενεργοποιείται ο TIMER_POTISMA_ENABLE
                                 (Επομένως μηδενίζει και ο TIMER_POTISMA)
  }

if (!FLOTER1 && FLOTER2)         Αν έχω floter1 και δεν έχω floter2.
{
    PUMP_RELAY=1;                Ενεργοποιείται η αντλία για γέμισμα δεξαμενής.
     strcpypgm2ram((char*)LCDText, "DEKSAMENH      "
                                                    "MISOGEMATH    ");
                                 LCDText[28]=(48+dec);
                                 LCDText[29]=(48+ypol);
     LCDUpdate();
    POTISMA_RELAY=0;             Απενεργοποιείται το πότισμα.
    TIMER_POTISMA_ENABLE=1;      Ενεργοποιείται ο TIMER_POTISMA_ENABLE
                                 (Επομένως ξεκινάει να μετράει και ο
                                                    TIMER_POTISMA)
   }


 if (!FLOTER2 && !FLOTER1)       Αν έχω floter1 και έχω floter2.
 {
     PUMP_RELAY=0;               Απενεργοποιείται η αντλία για γέμισμα δεξαμενής.
      strcpypgm2ram((char*)LCDText, "DEKSAMENH      "
                                                    "GEMATH        ");
                                 LCDText[28]=(48+dec);
                                 LCDText[29]=(48+ypol);
      LCDUpdate();
    TIMER_POTISMA_ENABLE=1;      Ενεργοποιείται ο TIMER_POTISMA_ENABLE
                                  (Επομένως ξεκινάει να μετράει και ο
                                                    TIMER_POTISMA)
 }
    if (TIMER_POTISMA>10)
     POTISMA_RELAY=1;            Ενεργοποιείται το πότισμα.

    if (TIMER_POTISMA>20)
     {
     POTISMA_RELAY=0;            Απενεργοποιείται το πότισμα.
     TIMER_POTISMA_ENABLE=0;     Απενεργοποιείται ο TIMER_POTISMA_ENABLE
```

(Επομένως μηδενίζει και ο TIMER_POTISMA)

```
  }

 }
///******************** TELOS PTYXIAKHS ********************///
//************************************************************//
itoa(*((WORD*)(&ADRESL)), AN0String);
   potensio=*((WORD*)(&ADRESL));
```

Με την εντολή itoa μετατρέπουμε έναν integer σε string.

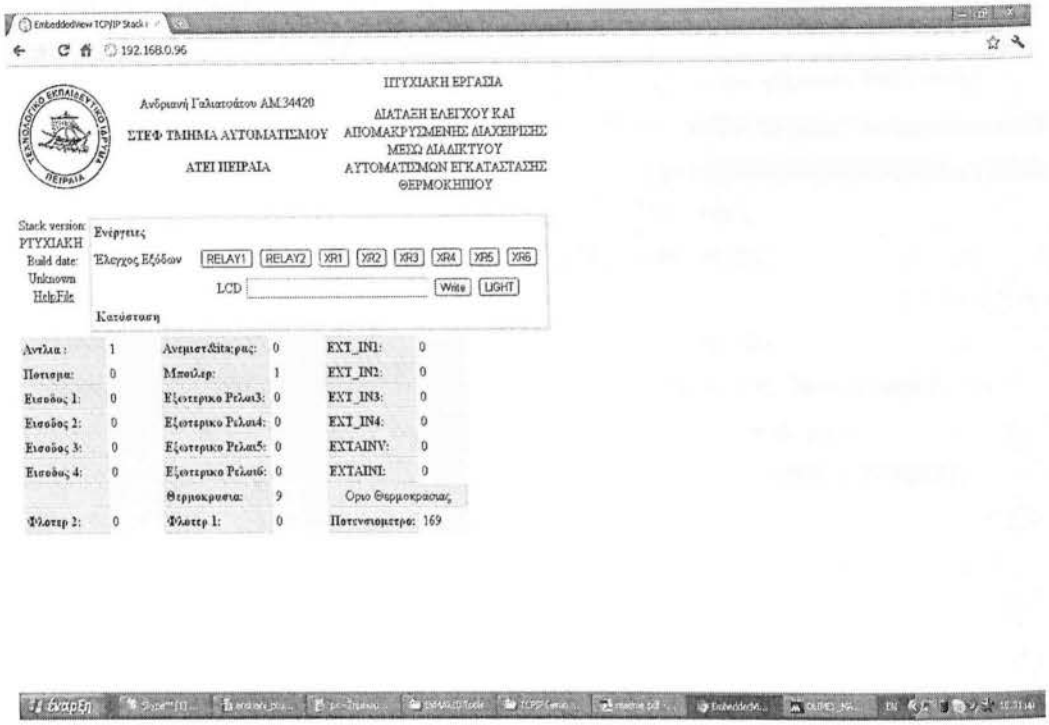
```
thermokrasia=Temperature;
itoa(Temperature, TEMPString)
```

Κεφάλαιο 5

Δημιουργία ιστοσελίδας για απομακρυσμένη διαχείριση μέσω διαδικτύου

Τρόπος δημιουργίας και λειτουργίας ιστοσελίδας

Η ιστοσελίδα μας έχει την εξής μορφή:



Πώς φτιάξαμε την ιστοσελίδα

Η ιστοσελίδα είναι βασισμένη σε τεχνολογία HTML και JavaScript. Τα στατικά στοιχεία της ιστοσελίδας είναι γραμμένα σε HTML ενώ τα στοιχεία της ιστοσελίδας που μεταβάλλονται οι τιμές τους, όπως ο πίνακας της ιστοσελίδας, είναι γραμμένα σε JavaScript έτσι ώστε οι μεταβολές των τιμών να φαίνονται σε πραγματικό χρόνο στην οθόνη μας.

Ο κώδικας φαίνεται αναλυτικά στο παράρτημα β στο τέλος στη σελίδα .

Βασικές οδηγίες:

1) Το προϊόν αυτό είναι βασισμένο στο MICROCHIP TCP/IP stack και εκεί έγκειται και η αξιοπιστία του κατά τη διάρκεια της λειτουργίας.

2) **Για να χειριστείτε τις εξόδους και να διαβάσετε τις εισόδους** υπάρχουν προς το παρόν 3 τρόποι:

α) Μέσω **HTTP** ανοίγοντας ένα browser (πχ Firefox) και δίνοντας την IP διεύθυνση του αναπτυξιακού. Αρχικά η διεύθυνση είναι προγραμματισμένη στο 192.168.0.95 αλλά θέλει προσοχή το ότι είναι ενεργοποιημένο το DCHP οπότε ο server μπορεί να προσδώσει στη συσκευή άλλη διεύθυνση. Προσοχή το HTTP δουλεύει ***μόνο εάν έχει γίνει login στο telnet interface***, για λόγους ασφαλείας επειδή προσωρινά λόγω ελλείψεως post requests δεν μπορεί να γίνει με ασφαλέστερο τρόπο η πρόσβαση στην ιστοσελίδα ελέγχου. Στην επόμενη έκδοση υλοποιείται η post method οπότε θα μπορεί να γίνει και ασφαλής πρόσβαση απευθείας στην ιστοσελίδα.

β) Μέσω **Telnet** εάν ανοίξετε ένα terminal ή γράψετε στο command των windows telnet 192.168.0.95 με username=admin και password=password (factory settings). Εκεί ακολουθείτε το μενού για να ελέγξετε τις 2 εξόδους (πατώντας '1' και μετά 'a' ή 'b'). Για να δούμε την κατάσταση των εισόδων και τη θερμοκρασία χώρου, στο αρχικό μενού πατάμε '2'.

γ) Μέσω **FTP** συνδεόμαστε με admin/password στον FTP server του EVMAX και στη συνέχεια μπορούμε να χρησιμοποιήσουμε την εντολή cd ως ακολούθως:

cd R1	αλλάζει την κατάσταση του ρελαί 1
cd R2	αλλάζει την κατάσταση του ρελαί 2
cd X1	αλλάζει την κατάσταση της εξόδου 1
cd X2	αλλάζει την κατάσταση της εξόδου 2
cd X3	αλλάζει την κατάσταση της εξόδου 3
cd X4	αλλάζει την κατάσταση της εξόδου 4
cd X5	αλλάζει την κατάσταση της εξόδου 5
cd status	επιστρέφει την κατάσταση όλων των εισόδων/εξόδων

3) **Για βασικό configuration** του board ακόμα και όταν αυτό δεν έχει σωστή IP διεύθυνση, κάνουμε τα εξής:

α) Συνδέουμε την σειριακή πόρτα του αναπτυξιακού με τον υπολογιστή μας με ένα καλώδιο null modem. (Σειριακό καλώδιο)

β) Ρυθμίζουμε ένα τερματικό (πχ Hyperterminal στα windows) ως εξής: 19200,8,N,1

γ) Πατάμε το μπουτόν 1 πάνω στο αναπτυξιακό και ενώ το κρατάμε πατημένο κάνουμε reset στο board με το μικρό κόκκινο κουμπί. Αφού αφήσουμε το reset, μετά από 1 δευτερόλεπτο αφήνουμε και το μπουτον 1 που κρατούσαμε πατημένο όλη αυτή την ώρα

δ) Θα εμφανιστεί μενού στο τερματικό μας από όπου μπορούμε να αλλάξουμε σειριακό αριθμό, όνομα συσκευής, IP, Gateway,Subnet Mask, DNS, DCHP enable status, MAC Address etc.

ε) Το password αλλάζει από το telnet interface

4) **Για να κάνετε τη δική σας ιστοσελίδα:**

α) Φτιάξτε ένα φάκελο στον οποίο θα βάλετε όλα όσα αφορούν την ιστοσελίδα, με ένα αρχείο να ονομάζεται index.htm να υπάρχει ως εναρκτήριο αρχείο της σελίδας. Φροντίστε να μην ξεπεράσετε τα 64KB

β) Ανοίξτε ένα command παράθυρο που να δείχνει στο φάκελο που υπάρχει ο φάκελος των αρχείων της HTML. Στον ίδιο αυτό φάκελο πρέπει να έχετε τοποθετήσει και το αρχείο MPFS.exe

γ) Εάν ο φάκελος που έχετε βάλει τα αρχεία της ιστοσελίδας ονομάζεται "WebSiteFolder" πρέπει να τρέξετε στο command line την εφαρμογή MPFS.exe συντεταγμένη ως εξής: "MPFS.exe WebSiteFolder test.bin"

δ) Τώρα όλα τα αρχεία που έχετε μέσα στο φάκελο της ιστοσελίδας συμπιέζονται στο αρχείο test.bin και είναι έτοιμα να κατέβουν στην EEPROM της συσκευής.

ε) Κάνουμε FTP 192.168.0.95 και με admin/password συνδεόμαστε με τη συσκευή. Στη συνέχεια δίνουμε "put test.bin" και περιμένουμε να ολοκληρωθεί το κατέβασμα.

ζ) Εάν θέλουμε έχουμε τη δυνατότητα να κατεβάσουμε τη σελίδα και μέσω Hyperterminal, εάν κάνουμε "send file" έχοντας ορίσει ως πρωτόκολλο αποστολής X-Modem. Το πρόγραμμα θα μας ζητήσει να διαλέξουμε αρχείο που θα κατέβει, οπότε εμείς επιλέγουμε το test.bin

ΠΑΡΑΡΤΗΜΑ Α

Κώδικας (προγράμματος)

Ο κώδικας είναι γραμμένος σε γλώσσα C και είναι ο εξής:

```
#define THIS_IS_STACK_APPLICATION
#define VERSION              "PTYXIAKH"//"v4.02"      // TCP/IP stack version
#define BAUD_RATE      (19200)              // bps

/************4 EKSODOI***************/

#define PUMP_RELAY INREL1_IO
#define POTISMA_RELAY INREL2_IO
#define ANEMISTHRAS_RELAY EXTREL1_IO
#define BOILER_RELAY EXTREL2_IO

/*********2EISODOI*****************/

#define FLOTER1 BUTTON0_IO
#define FLOTER2 BUTTON1_IO


// These headers must be included for required defs.
#include "TCPIP Stack/TCPIP.h"
#include "test.h"

// This is used by other stack elements.
// Main application must define this and initialize it with proper values.
APP_CONFIG AppConfig;
BYTE myDHCPBindCount = 0xFF;
BYTE AN0String[8];
char TEMPString[8];
BYTE LEVString[8];
BYTE VString[8];
BYTE TIMER_POTISMA_ENABLE;
int TIMER_POTISMA;
int dec;
int ypol;
```

```
int potensio;
int thermokrasia;


#if !defined(STACK_USE_DHCP_CLIENT)
	#define DHCPBindCount		(0xFF)
#endif


// Set configuration fuses
#if defined(__18CXX)
	#if defined(__18F8722)
		// PICDEM HPC Explorer board
		#pragma config OSC=HSPLL, FCMEN=OFF, IESO=OFF, PWRT=OFF,
WDT=OFF, LVP=OFF
	#elif defined(_18F8722)		// HI-TECH PICC-18 compiler
		// PICDEM HPC Explorer board
		__CONFIG(1, HSPLL);
		__CONFIG(2, WDTDIS);
		__CONFIG(3, MCLREN);
		__CONFIG(4, XINSTDIS & LVPDIS);
	#elif defined(__18F87J10) || defined(__18F86J15) || defined(__18F86J10) ||
defined(__18F85J15) || defined(__18F85J10) || defined(__18F67J10) ||
defined(__18F66J15) || defined(__18F66J10) || defined(__18F65J15) ||
defined(__18F65J10)
		// PICDEM HPC Explorer board
		#pragma config XINST=OFF, WDTEN=OFF, FOSC2=ON, FOSC=HSPLL
	#elif defined(__18F97J60) || defined(__18F96J65) || defined(__18F96J60) ||
defined(__18F87J60) || defined(__18F86J65) || defined(__18F86J60) ||
defined(__18F67J60) || defined(__18F66J65) || defined(__18F66J60)
		// PICDEM.net 2 and PICDEM.net 2 mini boards
		#pragma config XINST=OFF, WDT=OFF, FOSC2=ON, FOSC=HSPLL,
ETHLED=ON
	#elif defined(_18F97J60) || defined(_18F96J65) || defined(_18F96J60) ||
defined(_18F87J60) || defined(_18F86J65) || defined(_18F86J60) || defined(_18F67J60)
|| defined(_18F66J65) || defined(_18F66J60)
		// PICDEM.net 2 board with HI-TECH PICC-18 compiler
		__CONFIG(1, WDTDIS & XINSTDIS);
		__CONFIG(2, HSPLL);
	#elif defined(__18F4620)
		// PICDEM.net board
```

```
            #pragma config OSC=HS, WDT=OFF, MCLRE=ON, PBADEN=OFF,
LVP=OFF, XINST=OFF
            // PICDEM Z board
            //#pragma config OSC=HSPLL, WDT=OFF, MCLRE=ON, PBADEN=OFF,
LVP=OFF, XINST=OFF
        #elif defined(__18F452)
            // PICDEM.net board
            #pragma config OSC=HSPLL, WDT=OFF, LVP=OFF
        #endif
#elif defined(__PIC24F__)
        // Explorer 16 board
        _CONFIG2(FNOSC_PRIPLL & POSCMOD_XT)          // Primary XT OSC with 4x
PLL
        _CONFIG1(JTAGEN_OFF & FWDTEN_OFF)            // JTAG off, watchdog timer
off
#elif defined(__dsPIC33F__) || defined(__PIC24H__)
        // Explorer 16 board
        _FOSCSEL(FNOSC_PRIPLL)                  // PLL enabled
        _FOSC(OSCIOFNC_OFF & POSCMD_XT)         // XT Osc
        _FWDT(FWDTEN_OFF)                       // Disable Watchdog timer
        // JTAG should be disabled as well
#elif defined(__dsPIC30F__)
        // dsPICDEM 1.1 board
        _FOSC(XT_PLL16)                             // XT Osc + 16X PLL
        _FWDT(WDT_OFF)                              // Disable Watchdog timer
        _FBORPOR(MCLR_EN & PBOR_OFF & PWRT_OFF)
#endif


// Private helper functions.
// These may or may not be present in all applications.
static void InitAppConfig(void);
static void InitializeBoard(void);
static void ProcessIO(void);
#if defined(STACK_USE_SMTP_CLIENT)
static void SMTPDemo(void);
#endif
#if defined(STACK_USE_ICMP_CLIENT)
static void PingDemo(void);
#endif
extern BOOL TelnetAuthenticated;//coskon password authentication flag
static void DisplayIPValue(IP_ADDR IPVal);
```

```
static void FormatNetBIOSName(BYTE Name[16]);

#if defined(STACK_USE_UART)
        static void SetConfig(void);
#endif


// NOTE: Several PICs, including the PIC18F4620 revision A3 have a RETFIE FAST/MOVFF
bug
// The interruptlow keyword is used to work around the bug when using C18
extern void MACISR(void);
#if defined(HI_TECH_C)
void interrupt low_priority LowISR(void)
#else
#pragma interruptlow LowISR
void LowISR(void)
#endif
{
#ifdef __18CXX
   TickUpdate();
#endif

#if defined(STACK_USE_SLIP)
   MACISR();
#endif
}

#if defined(__18CXX) && !defined(HI_TECH_C)
#pragma code lowVector=0x18
void LowVector(void){_asm goto LowISR _endasm}
#pragma code // Return to default code section
#endif



// Main application entry point.
#ifdef __C30__
int main(void)
#else
void main(void)
```

```
#endif
{
    static TICK t = 0;
int  poten_val;
int  therm_val;
int proxeiro;

    // Initialize any application specific hardware.
    InitializeBoard();

#ifdef USE_LCD
        // Initialize and display the stack version on the LCD
        LCDInit();
        DelayMs(100);

        strcpypgm2ram((char*)LCDText, "TEI PEIRAIA    "
                                                    " A.Galiatsatou  ");
        LCDUpdate();

#endif

    // Initialize all stack related components.
    // Following steps must be performed for all applications using
    // the Microchip TCP/IP Stack.
    TickInit();

#if defined(STACK_USE_MPFS)
    // Initialize Microchip File System module
    MPFSInit();
#endif

    // Initialize Stack and application related NV variables into AppConfig.
    InitAppConfig();
    // Initiates board setup process if button is depressed
        // on startup

    if(BUTTON0_IO == 0u)
    {
                #if defined(MPFS_USE_EEPROM) && defined(STACK_USE_MPFS)
                // Invalidate the EEPROM contents if BUTTON0 is held down for more than
4 seconds
                TICK StartTime = TickGet();
```

```
            while(BUTTON0_IO == 0u)
            {
                if(TickGet() - StartTime > 4*TICK_SECOND)
                {
                  XEEBeginWrite(0x0000);
                  XEEWrite(0xFF);
                  XEEEndWrite();
                    #if defined(STACK_USE_UART)
                    putrsUART("\r\n\r\nBUTTON0 held for more than 4
seconds. EEPROM contents erased.\r\n\r\n");
                    #endif
                    LED0_TRIS = 0;
                    LED1_TRIS = 0;
                    LED2_TRIS = 0;
                    LCDLIGHT_TRIS = 0;
                    LED0_IO = 1;
                    LED1_IO = 1;
                    LED2_IO = 1;
                    LCDLIGHT_IO = 1;
                    while((LONG)(TickGet() - StartTime) <=
(LONG)(9*TICK_SECOND/2));
                    Reset();
                    break;
                }
            }
            #endif

            #if defined(STACK_USE_UART)
    SetConfig();
            #endif
  }
  //SetConfig();

      // Initialize core stack layers (MAC, ARP, TCP, UDP)
  StackInit();

#if defined(STACK_USE_HTTP_SERVER)
  HTTPInit();
#endif
```

```
#if     defined(STACK_USE_FTP_SERVER)     &&     defined(MPFS_USE_EEPROM)     &&
defined(STACK_USE_MPFS)
   FTPInit();
#endif

#if defined(STACK_USE_SNMP_SERVER)
        SNMPInit();
#endif

#if defined(STACK_USE_DHCP_CLIENT) || defined(STACK_USE_IP_GLEANING)
   if(!AppConfig.Flags.bIsDHCPEnabled)
   {
      // Force IP address display update.
      myDHCPBindCount = 1;
#if defined(STACK_USE_DHCP_CLIENT)
      DHCPDisable();
#endif
   }
#endif

                // Ivan - print out the IP adress
                #if defined(STACK_USE_UART)
                        putrsUART((ROM char*)"\r\nNew IP Addres: ");
                        DisplayIPValue(AppConfig.MyIPAddr);  // Print to UART
                        putrsUART((ROM char*)"\r\n");
                #endif
 // Once all items are initialized, go into infinite loop and let
 // stack items execute their tasks.
 // If application needs to perform its own task, it should be
 // done at the end of while loop.
 // Note that this is a "co-operative mult-tasking" mechanism
 // where every task performs its tasks (whether all in one shot
 // or part of it) and returns so that other tasks can do their
 // job.
 // If a task needs very long time to do its job, it must be broken
 // down into smaller pieces so that other tasks can have CPU time.
      INREL1_TRIS=0;
      INREL2_TRIS=0;
      EXTREL1_TRIS=0;
      EXTREL2_TRIS=0;
      EXTREL3_TRIS=0;
      EXTREL4_TRIS=0;
```

```
    EXTREL5_TRIS=0;
    EXTREL6_TRIS=0;
    INREL1_IO=0;
    INREL2_IO=0;
    EXTREL1_IO=0;
    EXTREL2_IO=0;
    EXTREL3_IO=0;
    EXTREL4_IO=0;
    EXTREL5_IO=0;
    EXTREL6_IO=0;

    IN1_TRIS=1;
    IN2_TRIS=1;
    IN3_TRIS=1;
    IN4_TRIS=1;
    EXTIN1_TRIS=1;
    EXTIN2_TRIS=1;
    EXTIN3_TRIS=1;
    EXTIN4_TRIS=1;

  while(1)
  {
    // Blink LED0 (right most one) every second.
    if(TickGet() - t >= TICK_SECOND)
    {
      t = TickGet();
      LED0_IO ^= 1;
///******************** ARXH PTYXIAKHS **********************///

//ANDR****************EKTELEITAI KATHE DEFTEROLEPTO************//

if (TIMER_POTISMA_ENABLE)TIMER_POTISMA++;
if (!TIMER_POTISMA_ENABLE)TIMER_POTISMA=0;

dec=TIMER_POTISMA/10;
ypol=TIMER_POTISMA%10;

/********dokimes********/
/*
if(potensio>450)
EXTREL3_IO=1;
else
```

```
 EXTREL3_IO=0;
 */

 /*
  if (thermokrasia>potensio)
     {EXTREL2_IO=1;}
    if (thermokrasia<=potensio)
     {EXTREL2_IO=0;}
 */
 /***********************/



 /************************THERMOKHPIO*************************/

  if (thermokrasia>((potensio/10)+1))
     {ANEMISTHRAS_RELAY=1;}
  if (thermokrasia<=(potensio/10))
     {ANEMISTHRAS_RELAY=0;}
  if (thermokrasia<((potensio/10)-10))
     {BOILER_RELAY=1;}
  if (thermokrasia>=(potensio/10))
     {BOILER_RELAY=0;}

 /*************************************************************/

 ///*********************AYTOMATO POTISMA*******************///

 if (FLOTER1 && FLOTER2)
 {
        PUMP_RELAY=1;
     strcpypgm2ram((char*)LCDText, "DEKSAMENH      "
                                                    "GEMIZEI       ");
                                              LCDText[28]=(48+dec);
                 LCDText[29]=(48+ypol);
        LCDUpdate();
        POTISMA_RELAY=0;
   TIMER_POTISMA_ENABLE=0;
  }

if (!FLOTER1 && FLOTER2)
{
```

```
      PUMP_RELAY=1;
   strcpypgm2ram((char*)LCDText, "DEKSAMENH     "
                                                    "MISOGEMATH    ");
                                                LCDText[28]=(48+dec);
                LCDText[29]=(48+ypol);
       LCDUpdate();
       POTISMA_RELAY=0;
  TIMER_POTISMA_ENABLE=1;
  }

if (!FLOTER2 && !FLOTER1)
{
   PUMP_RELAY=0;
    TIMER_POTISMA_ENABLE=1;
    strcpypgm2ram((char*)LCDText, "DEKSAMENH     "
                                                    "GEMATH       ");
                    LCDText[28]=48+dec;
                LCDText[29]=48+ypol;
        LCDUpdate();
 }

   if (TIMER_POTISMA>10)
     POTISMA_RELAY=1;
   if (TIMER_POTISMA>20)
     {
      POTISMA_RELAY=0;
      TIMER_POTISMA_ENABLE=0;
     }
  }

 ///********************  TELOS   PTYXIAKHS  ******************///
 //************************************************************//


      // This task performs normal stack task including checking
      // for incoming packet, type of packet and calling
      // appropriate stack entity to process it.
      StackTask();

  #if defined(STACK_USE_HTTP_SERVER)
      // This is a TCP application.  It listens to TCP port 80
      // with one or more sockets and responds to remote requests.
```

```
//if(TelnetAuthenticated==1)//coskon password authentication flag)
        HTTPServer();
#endif

#if     defined(STACK_USE_FTP_SERVER)     &&     defined(MPFS_USE_EEPROM)     &&
defined(STACK_USE_MPFS)
        FTPServer();
#endif

#if defined(STACK_USE_SNMP_SERVER)
                SNMPTask();
#endif

#if defined(STACK_USE_ANNOUNCE)
                DiscoveryTask();
#endif

#if defined(STACK_USE_NBNS)
                NBNSTask();
#endif

#if defined(STACK_USE_DHCP_SERVER)
                DHCPServerTask();
#endif

#if defined(STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE)
                GenericTCPClient();
#endif

#if defined(STACK_USE_GENERIC_TCP_SERVER_EXAMPLE)
                GenericTCPServer();
#endif

#if defined(STACK_USE_TELNET_SERVER)
                TelnetTask();
#endif

#if defined(STACK_USE_REBOOT_SERVER)
                RebootTask();
#endif

#if defined(STACK_USE_UDP_PERFORMANCE_TEST)
```

```
            UDPPerformanceTask();
#endif

#if defined(STACK_USE_TCP_PERFORMANCE_TEST)
            TCPPerformanceTask();
#endif

#if defined(STACK_USE_SMTP_CLIENT)
            SMTPTask();
            SMTPDemo();
#endif

#if defined(STACK_USE_ICMP_CLIENT)
            PingDemo();
#endif

    // Add your application specific tasks here.
    ProcessIO();


    // For DHCP information, display how many times we have renewed the IP
    // configuration since last reset.
    if(DHCPBindCount != myDHCPBindCount)
    {
      myDHCPBindCount = DHCPBindCount;

                #if defined(STACK_USE_UART)
                    putrsUART((ROM char*)"New IP Addres: ");
                #endif

      DisplayIPValue(AppConfig.MyIPAddr);  // Print to UART

                #if defined(STACK_USE_UART)
                    putrsUART((ROM char*)"\r\n");
                #endif

                #if defined(STACK_USE_ANNOUNCE)
                    AnnounceIP();
                #endif
    }
  }
}
```

```
static void DisplayIPValue(IP_ADDR IPVal)
{
//      printf("%u.%u.%u.%u", IPVal.v[0], IPVal.v[1], IPVal.v[2], IPVal.v[3]);
  BYTE IPDigit[4];
      BYTE i;
#ifdef USE_LCD
      BYTE j;
      BYTE LCDPos=16;
#endif

      for(i = 0; i < sizeof(IP_ADDR); i++)
      {
        uitoa((WORD)IPVal.v[i], IPDigit);

          #if defined(STACK_USE_UART)
              putsUART(IPDigit);
          #endif

          #ifdef USE_LCD
              for(j = 0; j < strlen((char*)IPDigit); j++)
              {
                  LCDText[LCDPos++] = IPDigit[j];
              }
              if(i == sizeof(IP_ADDR)-1){
                  break;
              }
              LCDText[LCDPos++] = '.';
          #else
              if(i == sizeof(IP_ADDR)-1)
                  break;
          #endif

          #if defined(STACK_USE_UART)
              while(BusyUART());
              WriteUART('.');
          #endif
    }

    #ifdef USE_LCD
        if(LCDPos < 32)
```

```
                    LCDText[LCDPos] = 0;
        strcpypgm2ram((char*)LCDText, "TEI PEIRAIA     "
                                                    " A.Galiatsatou  ");
        LCDUpdate();
        #endif
}


static void ProcessIO(void)
{
#ifdef __C30__
   // Convert potentiometer result into ASCII string
   uitoa((WORD)ADC1BUF0, AN0String);
#else
   // AN0 should already be set up as an analog input
   /*ADCON0bits.GO = 1;

   // Wait until A/D conversion is done
   while(ADCON0bits.GO);

        // AD converter errata work around (ex: PIC18F87J10 A2)
        #if !defined(__18F452)
        PRODL = ADCON2;
        ADCON2bits.ADCS0 = 1;
        ADCON2bits.ADCS1 = 1;
        ADCON2 = PRODL;
        #endif


    // Convert 10-bit value into ASCII string

    uitoa(*((WORD*)(&ADRESL)), AN0String);*/
        int Temperature, tmp;


        //---------------- ANTR ----------------------
        // ADC chanel2
        ADCON0bits.CHS0 = 0;
        ADCON0bits.CHS1 = 1;
        ADCON0bits.CHS2 = 0;
        ADCON0bits.CHS3 = 0;
```

```
    // AN0 should already be set up as an analog input
    ADCON0bits.GO = 1;

    // Wait until A/D conversion is done
    while(ADCON0bits.GO);

    // Convert 10-bit value into ASCII string
    itoa(*((WORD*)(&ADRESL)), AN0String);
   potensio=*((WORD*)(&ADRESL));



        // Turn OFF ADC
        // ADCON0bits.ON = 0;

//---------------- Analog Voltage 0-10 Volt ------------------
        // ADC chanel14
        ADCON0bits.CHS0 = 0;
        ADCON0bits.CHS1 = 1;
        ADCON0bits.CHS2 = 1;
        ADCON0bits.CHS3 = 1;

    // AN0 should already be set up as an analog input
    ADCON0bits.GO = 1;

    // Wait until A/D conversion is done
    while(ADCON0bits.GO);
    // Convert 10-bit value into ASCII string
    itoa(*((WORD*)(&ADRESL)), VString);
        // Turn OFF ADC
        // ADCON0bits.ON = 0;

//---------------- Analog Current 4-20mA ------------------
        // ADC chanel5
        ADCON0bits.CHS0 = 1;
        ADCON0bits.CHS1 = 1;
        ADCON0bits.CHS2 = 1;
        ADCON0bits.CHS3 = 1;

    // AN0 should already be set up as an analog input
    ADCON0bits.GO = 1;
```

```
// Wait until A/D conversion is done
while(ADCON0bits.GO);
// Convert 10-bit value into ASCII string
itoa(*((WORD*)(&ADRESL)), LEVString);
      // Turn OFF ADC
      // ADCON0bits.ON = 0;

      //--------------- TEMP -------------------------

      // ADC chanel3
      ADCON0bits.CHS0 = 1;
      ADCON0bits.CHS1 = 1;
      ADCON0bits.CHS2 = 0;
      ADCON0bits.CHS3 = 0;


      // Turn on ADC
      // ADCON0bits.ON = 1;

 // AN0 should already be set up as an analog input
 ADCON0bits.GO = 1;

 // Wait until A/D conversion is done
 while(ADCON0bits.GO);


 // Convert temperature result into ASCII string
      // Temperature = ((float)(ADRESL)*(3.3/1024.)-0.500)*100.;
 //
      tmp = *((WORD*)(&ADRESL));

      if(Temperature>512) {
            tmp = tmp - 512;
            tmp/=9;
            Temperature = 20 + tmp;
      }
      else {
            tmp = 512 - tmp;
            tmp/=9;
            Temperature = 20 - tmp;
      }
```

```
        if(Temperature<0)
                Temperature = 0;

        //sprintf(TEMPString, "%3.1fC", Temperature);

        // Convert 10-bit value into ASCII string
    thermokrasia=Temperature;
        itoa(Temperature, TEMPString);
#endif
}
#if defined(STACK_USE_SMTP_CLIENT)
static void SMTPDemo(void)
{
        // Send an email once if someone pushes BUTTON0
        // This is a multi-part message example, where the message
        // body is dynamically generated and need not fit in RAM.
        // LED1 will be used as a busy indicator
        // LED2 will be used as a mail sent successfully indicator
        static enum
        {
                MAIL_HOME = 0,
                MAIL_BEGIN,
                MAIL_PUT_DATA,
                MAIL_PUT_DATA2,
                MAIL_SMTP_FINISHING,
                MAIL_DONE
        } MailState = MAIL_HOME;
        static BYTE *MemPtr;
        static TICK WaitTime;

        switch(MailState)
        {
                case MAIL_HOME:
                  if(BUTTON0_IO == 0u)
                      {
                              // Start sending an email
                              LED1_IO = 1;
                              MailState++;
                              LED2_IO = 0;
                      }
                      break;
```

```
            case MAIL_BEGIN:
                if(SMTPBeginUsage())
                {
                    // Note that these strings must stay allocated in
                    // memory until SMTPIsBusy() returns FALSE.  To
                    // guarantee that the C compiler does not reuse this
                    // memory, you must allocate the strings as static.

                    //static BYTE RAMStringServer[] = "mail";       //      SMTP
server address
                    static BYTE RAMStringTo[] = "joe@picsaregood.com";
                    //static  BYTE  RAMStringCC[]  =  "foo@picsaregood.com,
\"Jane Smith\" <jane.smith@picsaregood.com>";

                    //SMTPClient.Server.szRAM = RAMStringServer;
                    SMTPClient.To.szRAM = RAMStringTo;
                    //SMTPClient.CC.szRAM = RAMStringCC;
                    SMTPClient.From.szROM = (ROM BYTE*)"\"SMTP Service\"
<mchpboard@picsaregood.com>";
                    SMTPClient.ROMPointers.From = 1;
                    SMTPClient.Subject.szROM  =  (ROM  BYTE*)"Hello  world!
SMTP Test.";
                    SMTPClient.ROMPointers.Subject = 1;
                    SMTPSendMail();
                    MailState++;
                }
                break;

            case MAIL_PUT_DATA:
                // Check to see if a failure occured
                if(!SMTPIsBusy())
                {
                    // Finished sending mail
                    LED1_IO = 0;
                    MailState = MAIL_DONE;
                    WaitTime = TickGet();
                    LED2_IO = (SMTPEndUsage() == SMTP_SUCCESS);
                    break;
                }

                if(SMTPIsPutReady() >= 121u)
                {
```

```
                    SMTPPutROMString((ROM BYTE*)"Hello!\r\n\r\nThis mail
was automatically generated by EmbeddedView TCP/IP Stack " VERSION ".\r\n\r\nThe
following is a snapshot of RAM:\r\n");
                    SMTPFlush();

                    MemPtr = 0x0000;
                    MailState++;
                }
                break;

        case MAIL_PUT_DATA2:
                // Check to see if a failure occured
                if(!SMTPIsBusy())
                {
                        // Finished sending mail
                        LED1_IO = 0;
                        MailState = MAIL_DONE;
                        WaitTime = TickGet();
                        LED2_IO = (SMTPEndUsage() == SMTP_SUCCESS);
                        break;
                }

                if(SMTPIsPutReady() >= 75u)
                {
                        BYTE i, c;
                        WORD_VAL w;

                        // Write line address
                        w.Val = (WORD)MemPtr;
                        SMTPPut(btohexa_high(w.v[1]));
                        SMTPPut(btohexa_low(w.v[1]));
                        SMTPPut(btohexa_high(w.v[0]));
                        SMTPPut(btohexa_low(w.v[0]));
                        SMTPPut(' ');

                        // Write data bytes in hex
                        for(i = 0; i < 16u; i++)
                        {
                                SMTPPut(' ');
                                c = *MemPtr++;
                                SMTPPut(btohexa_high(c));
                                SMTPPut(btohexa_low(c));
```

```
                        if(i == 7u)
                                SMTPPut(' ');
                }

                SMTPPut(' ');
                SMTPPut(' ');

                // Write data bytes in ASCII
                MemPtr -= 16;
                for(i = 0; i < 16u; i++)
                {
                        c = *MemPtr++;
                        if(c < ' ' || c > '~')
                                c = '.';
                        SMTPPut(c);

                        if(i == 7u)
                                SMTPPut(' ');
                }

                SMTPPut('\r');
                SMTPPut('\n');
                SMTPFlush();

                // Make sure not to read from memory above address
0x0E7F.
                // Doing so would disrupt volatile pointers, ERDPT, FSR0,
FSR1, FSR2, etc.
                if((WORD)MemPtr >= 0xE7Fu)
                {
                        SMTPPutDone();
                        MailState++;
                }
        }
        break;

    case MAIL_SMTP_FINISHING:
        // Check to see if we are done communicating with the SMTP
server
        if(!SMTPIsBusy())
        {
                // Finished sending mail
```

```
                        LED1_IO = 0;
                        MailState = MAIL_DONE;
                        WaitTime = TickGet();
                        LED2_IO = (SMTPEndUsage() == SMTP_SUCCESS);
                    }
                    break;

            case MAIL_DONE:
                    // Wait for the user to release BUTTON0 and for at
                    // least 1 second to pass before allowing another
                    // email to be sent.  This is merely to prevent
                    // accidental flooding of email boxes while
                    // developing code.  Your application may wish to
                    // remove this.
                    if(BUTTON0_IO)
                    {
                        if(TickGet() - WaitTime > TICK_SECOND)
                            MailState = MAIL_HOME;
                    }
                    break;
        }
}
/*
static void SMTPDemo(void)
{
    // Send an email once if someone pushes BUTTON0
    // This is a simple message example, where the message
    // body must already be in RAM.
    // LED1 will be used as a busy indicator
    // LED2 will be used as a mail sent successfully indicator
    static enum
    {
        MAIL_HOME = 0,
        MAIL_BEGIN,
        MAIL_SMTP_FINISHING,
        MAIL_DONE
    } MailState = MAIL_HOME;
    static TICK WaitTime;

    switch(MailState)
    {
        case MAIL_HOME:
```

```
            if(BUTTON0_IO == 0u)
                {
                    // Start sending an email
                    LED1_IO = 1;
                    MailState++;
                    LED2_IO = 0;
                }
                break;

            case MAIL_BEGIN:
                if(SMTPBeginUsage())
                {
                    // Note that these strings must stay allocated in
                    // memory until SMTPIsBusy() returns FALSE.  To
                    // guarantee that the C compiler does not reuse this
                    // memory, you must allocate the strings as static.

                    //static BYTE RAMStringServer[] = "mail";       //      SMTP
server address
                    static BYTE RAMStringTo[] = "joe@picsaregood.com";
                    //static  BYTE  RAMStringCC[]  =  "foo@picsaregood.com,
\"Jane Smith\" <jane.smith@picsaregood.com>";
                    //static BYTE RAMStringBCC[] = "";
                    static  BYTE  RAMStringBody[]  =  "Message  generated  by
stack " VERSION " \r\n\r\nButtons: 3210";
                    RAMStringBody[sizeof(RAMStringBody)-2]     =     '0'     +
BUTTON0_IO;
                    RAMStringBody[sizeof(RAMStringBody)-3]     =     '0'     +
BUTTON1_IO;
                    RAMStringBody[sizeof(RAMStringBody)-4]     =     '0'     +
EXTIN1_IO;
                    RAMStringBody[sizeof(RAMStringBody)-5]     =     '0'     +
EXTIN2_IO;

                    //SMTPClient.Server.szRAM = RAMStringServer;
                    SMTPClient.To.szRAM = RAMStringTo;
                    SMTPClient.From.szROM       =       "\"SMTP       Service\"
<mchpboard@picsaregood.com>";
                    SMTPClient.ROMPointers.From = 1;
                    SMTPClient.Subject.szROM = "Hello world!  SMTP Test.";
                    SMTPClient.ROMPointers.Subject = 1;
                    SMTPClient.Body.szRAM = RAMStringBody;
```

```
                        SMTPSendMail();
                        MailState++;
                    }
                    break;

                case MAIL_SMTP_FINISHING:
                    if(!SMTPIsBusy())
                    {
                        // Finished sending mail
                        LED1_IO = 0;
                        MailState++;
                        WaitTime = TickGet();
                        LED2_IO = (SMTPEndUsage() == SMTP_SUCCESS);
                    }
                    break;

                case MAIL_DONE:
                    // Wait for the user to release BUTTON0 and for at
                    // least 1 second to pass before allowing another
                    // email to be sent.  This is merely to prevent
                    // accidental flooding of email boxes while
                    // developing code.  Your application may wish to
                    // remove this.
                    if(BUTTON0_IO)
                    {
                        if(TickGet() - WaitTime > TICK_SECOND)
                            MailState = MAIL_HOME;
                    }
                    break;
        }
}
*/
#endif //#if defined(STACK_USE_SMTP_CLIENT)


// ICMP Echo (Ping) example code
#if defined(STACK_USE_ICMP_CLIENT)
static void PingDemo(void)
{
    static enum
    {
        SM_HOME = 0,
```

```
            SM_GET_RESPONSE
        } PingState = SM_HOME;
        static TICK Timer;
        SHORT ret;
        IP_ADDR RemoteIP;

        switch(PingState)
        {
                case SM_HOME:
                        // Send a ping request out if the user pushes EXTIN2 (left-most
one)
                        if(EXTIN2_IO == 0)
                        {
                                // Don't ping flood: wait at least 1 second between ping
requests
                                if((DWORD)(TickGet()-Timer) > 1u*TICK_SECOND)
                                {
                                        // Obtain ownership of the ICMP module
                                        if(ICMPBeginUsage())
                                        {
                                                Timer = TickGet();
                                                PingState = SM_GET_RESPONSE;

                                                // Send the ping request to 4.78.194.159
(ww1.microchip.com)
                                                RemoteIP.v[0] = 4;
                                                RemoteIP.v[1] = 78;
                                                RemoteIP.v[2] = 194;
                                                RemoteIP.v[3] = 159;
                                                ICMPSendPing(RemoteIP.Val);
                                        }
                                }
                        }
                        break;

                case SM_GET_RESPONSE:
                        // Get the status of the ICMP module
                        ret = ICMPGetReply();
                        if(ret == -2)
                        {
                                // Do nothing: still waiting for echo
                                break;
```

```
                        }
                        else if(ret == -1)
                        {
                                // Request timed out
                                #if defined(USE_LCD)
                                memcpypgm2ram((void*)&LCDText[16],    (ROM    void
*)"Ping timed out", 15);
                                LCDUpdate();
                                #endif
                                PingState = SM_HOME;
                        }
                        else
                        {
                                // Echo received.  Time elapsed is stored in ret (units of
TICK).
                                #if defined(USE_LCD)
                                memcpypgm2ram((void*)&LCDText[16],    (ROM    void
*)"Reply: ", 7);
                                uitoa(ret, &LCDText[16+7]);
                                strcatpgm2ram((char*)&LCDText[16+7],          (ROM
char*)"ms");
                                LCDUpdate();
                                #endif
                                PingState = SM_HOME;
                        }

                        // Finished with the ICMP module, release it so other apps can
begin using it
                        ICMPEndUsage();
                        break;
        }
}
#endif  //#if defined(STACK_USE_ICMP_CLIENT)


// CGI Command Codes
#define CGI_CMD_DIGOUT    (0)
#define CGI_CMD_LCDOUT    (1)
#define CGI_CMD_RECONFIG  (2)

// CGI Variable codes. - There could be 00h-FFh variables.
// NOTE: When specifying variables in your dynamic pages (.cgi),
```

```
//      use the hexadecimal numbering scheme and always zero pad it
//      to be exactly two characters.  Eg: "%04", "%2C"; not "%4" or "%02C"
#define VAR_IN1                                 (0x00) // DigitalInput1
#define VAR_IN2                                 (0x01) // DigitalInput2
#define VAR_IN3                                 (0x10) // DigitalInput3
#define VAR_IN4                                 (0x11) // DigitalInput4
#define VAR_EXTIN1                      (0x21) // ExternalDigitalInput1
#define VAR_EXTIN2                      (0x22) // ExternalDigitalInput2
#define VAR_EXTIN3                      (0x23) // ExternalDigitalInput3
#define VAR_EXTIN4                      (0x24) // ExternalDigitalInput4
#define VAR_INREL1                      (0x13) // Onboard RELAY1
#define VAR_INREL2                      (0x14) // Onboard RELAY2
#define VAR_EXTREL1                     (0x27) // ExternalRELAY1
#define VAR_EXTREL2                     (0x28) // ExternalRELAY2
#define VAR_EXTREL3                     (0x29) // ExternalRELAY3
#define VAR_EXTREL4                     (0x2A) // ExternalRELAY4
#define VAR_EXTREL5                     (0x2B) // ExternalRELAY5
#define VAR_EXTREL6                     (0x2C) // ExternalRELAY6
#define VAR_BUTTON1         (0x04) // Button1 on the Maxi board
#define VAR_BUTTON2         (0x0D) // Button2 on the Maxi board
#define VAR_EXTAINV                     (0x25) // Analog Input Voltage
#define VAR_EXTAINI                     (0x26) // Analog Input Current
#define VAR_ANAIN_AN0       (0x02)  // Analog Inputs TEMPERATURE
#define VAR_ANAIN_AN1       (0x03) // Analog Inputs POTENSIOMETER
#define VAR_ANAIN_AN14      (0x34) // Analog Inputs 0-10Volt
#define VAR_ANAIN_AN15      (0x35) // Analog Inputs 4-20mA Loop


#define VAR_LED4                        (0x12)
#define VAR_LED7                        (0x15)
#define VAR_DIGIN2          (0x0E) //
#define VAR_DIGIN3          (0x0F) //
#define VAR_STACK_VERSION (0x16) // Stack constants
#define VAR_STACK_DATE                  (0x17)
#define VAR_STROUT_LCD      (0x05) // LCD Display output
#define VAR_MAC_ADDRESS     (0x06)// Stack configuration variables
#define VAR_SERIAL_NUMBER  (0x07)
#define VAR_IP_ADDRESS      (0x08)
#define VAR_SUBNET_MASK     (0x09)
#define VAR_GATEWAY_ADDRESS (0x0A)
#define VAR_DHCP            (0x0B)
#define VAR_DHCP_TRUE       (0x0B)
```

```
#define VAR_DHCP_FALSE      (0x0C)


// CGI Command codes (CGI_CMD_DIGOUT).
// Should be a one digit numerical value
#define CMD_LED2                    (0x1)
#define CMD_INREL1                  (0x2)
#define CMD_INREL2                  (0x3)
#defineCMD_LCDLIGHT                     (0x4)
#define CMD_EXTREL1                 (0x5)
#define CMD_EXTREL2                 (0x6)
#define CMD_EXTREL3                 (0x7)
#define CMD_EXTREL4                 (0x8)
#defineCMD_EXTREL5                  (0x9)
#define CMD_EXTREL6                 (0x0)


/*********************************************************************
 * Function:        void HTTPExecCmd(BYTE** argv, BYTE argc)
 *
 * PreCondition:    None
 *
 * Input:           argv       - List of arguments
 *                  argc       - Argument count.
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        This function is a "callback" from HTTPServer
 *                  task.  Whenever a remote node performs
 *                  interactive task on page that was served,
 *                  HTTPServer calls this functions with action
 *                  arguments info.
 *                  Main application should interpret this argument
 *                  and act accordingly.
 *
 *                  Following is the format of argv:
 *                  If HTTP action was : thank.htm?name=Joe&age=25
 *                    argv[0] => thank.htm
 *                    argv[1] => name
 *                    argv[2] => Joe
```

```
*                argv[3] => age
*                argv[4] => 25
*
*             Use argv[0] as a command identifier and rests
*             of the items as command arguments.
*
* Note:          THIS IS AN EXAMPLE CALLBACK.
********************************************************************/
#if defined(STACK_USE_HTTP_SERVER)

ROM char COMMANDS_OK_PAGE[] = "INDEX.CGI";
ROM char CONFIG_UPDATE_PAGE[] = "CONFIG.CGI";
ROM char CMD_UNKNOWN_PAGE[] = "INDEX.CGI";

// Copy string with NULL termination.
#define COMMANDS_OK_PAGE_LEN    (sizeof(COMMANDS_OK_PAGE))
#define CONFIG_UPDATE_PAGE_LEN  (sizeof(CONFIG_UPDATE_PAGE))
#define CMD_UNKNOWN_PAGE_LEN    (sizeof(CMD_UNKNOWN_PAGE))

void HTTPExecCmd(BYTE** argv, BYTE argc)
{
   BYTE command;
   BYTE var;
#if defined(ENABLE_REMOTE_CONFIG)
        DWORD_VAL dwVal;
   BYTE CurrentArg;
   WORD_VAL TmpWord;
#endif
   /*
    * Design your pages such that they contain command code
    * as a one character numerical value.
    * Being a one character numerical value greatly simplifies
    * the job.
    */
   command = argv[0][0] - '0';

   /*
    * Find out the cgi file name and interpret parameters
    * accordingly
    */
   switch(command)
   {
```

```
   case CGI_CMD_DIGOUT:     // ACTION=0
      /*
       * Identify the parameters.
       * Compare it in upper case format.
       */
      var = argv[1][0] - '0';
//putcUART(var+'0');//coskon
//putrsUART("*\n\r");//coskon

      switch(var)
      {
               case CMD_LCDLIGHT:  // LCD LIGHT
         // Toggle LCDLIGHT.
         LCDLIGHT_IO ^= 1;
         break;
               case CMD_INREL1:      // RELAY1
         // Toggle RELAY1.
         INREL1_IO ^= 1;
         break;
               case CMD_INREL2:      // RELAY2
         // Toggle RELAY2.
         INREL2_IO ^= 1;
         break;
      case CMD_EXTREL1:
         // Toggle ExtRElay1.
         EXTREL1_IO ^= 1;
         break;
      case CMD_EXTREL2:
         // Toggle ExtRElay2
         EXTREL2_IO ^= 1;
//putrsUART("2*\n\r");//coskon
         break;
               case CMD_EXTREL3:
         // Toggle ExtRElay3
         EXTREL3_IO ^= 1;
//putrsUART("3*\n\r");//coskon
         break;
               case CMD_EXTREL4:
         // Toggle ExtRElay4
         EXTREL4_IO ^= 1;
         break;
               case CMD_EXTREL5:
```

```
        // Toggle ExtRElay5
        EXTREL5_IO ^= 1;
        break;
                case CMD_EXTREL6:
        // Toggle ExtRElay6
        EXTREL6_IO ^= 1;
        break;
      }

      memcpypgm2ram((void*)argv[0],        (ROM        void*)COMMANDS_OK_PAGE,
COMMANDS_OK_PAGE_LEN);
      break;
#if defined(USE_LCD)
   case CGI_CMD_LCDOUT:    // ACTION=1
                if(argc > 2u)   // Text provided in argv[2]
                {
                        // Convert %20 to spaces, and other URL transformations
                        UnencodeURL(argv[2]);

                        // Write 32 received characters or less to LCDText
                        if(strlen((char*)argv[2]) < 32u)
                        {
                                memset(LCDText, ' ', 32);
                                strcpy((char*)LCDText, (char*)argv[2]);
                        }
                        else
                        {
                                memcpy(LCDText, (void*)argv[2], 32);
                        }
                        // Write LCDText to the LCD
                        LCDUpdate();
                }
                else                    // No text provided
                {
                        LCDErase();
                }
                memcpypgm2ram((void*)argv[0],   (ROM   void*)COMMANDS_OK_PAGE,
   COMMANDS_OK_PAGE_LEN);
        break;
   #endif
   #if defined(ENABLE_REMOTE_CONFIG)
   // Possibly useful code for remotely reconfiguring the board through
```

```
// HTTP
        case CGI_CMD_RECONFIG:     // ACTION=2
                // Loop through all variables that we've been given
                CurrentArg = 1;
                while(argc > CurrentArg)
                {
                        // Get the variable identifier (HTML "name"), and
                        // increment to the variable's value
                        TmpWord.byte.MSB = argv[CurrentArg][0];
                        TmpWord.byte.LSB = argv[CurrentArg++][1];
             var = hexatob(TmpWord);

             // Make sure the variable's value exists
             if(CurrentArg >= argc)
               break;

             // Take action with this variable/value
             switch(var)
             {
             case VAR_IP_ADDRESS:
             case VAR_SUBNET_MASK:
             case VAR_GATEWAY_ADDRESS:
               {
                        // Convert the returned value to the 4 octect
                        // binary representation
                             if(!StringToIPAddress(argv[CurrentArg],
(IP_ADDR*)&dwVal))
                               break;

                                    // Reconfigure the App to use the new values
                             if(var == VAR_IP_ADDRESS)
                             {
                                    // Cause the IP address to be rebroadcast
                                    // through Announce.c or the RS232 port since
                                    // we now have a new IP address
                                    if(dwVal.Val != *(DWORD*)&AppConfig.MyIPAddr)
                                          DHCPBindCount++;

                                       // Set the new address
                                 memcpy((void*)&AppConfig.MyIPAddr,      (void*)&dwVal,
sizeof(AppConfig.MyIPAddr));
                             }
```

```
                        else if(var == VAR_SUBNET_MASK)
                          memcpy((void*)&AppConfig.MyMask,        (void*)&dwVal,
sizeof(AppConfig.MyMask));
                        else if(var == VAR_GATEWAY_ADDRESS)
                          memcpy((void*)&AppConfig.MyGateway,     (void*)&dwVal,
sizeof(AppConfig.MyGateway));
                    }
                break;

            case VAR_DHCP:
              if(AppConfig.Flags.bIsDHCPEnabled)
              {
                    if(!(argv[CurrentArg][0]-'0'))
                    {
                        AppConfig.Flags.bIsDHCPEnabled = FALSE;
                    }
                  }
                  else
              {
                    if(argv[CurrentArg][0]-'0')
                    {
                                        AppConfig.MyIPAddr.Val = 0x00000000ul;
                        AppConfig.Flags.bIsDHCPEnabled = TRUE;
                            AppConfig.Flags.bInConfigMode = TRUE;
                        DHCPReset();
                    }
                  }
               break;
               }

                    // Advance to the next variable (if present)
                    CurrentArg++;
        }

              // Save any changes to non-volatile memory
          SaveAppConfig();


              // Return the same CONFIG.CGI file as a result.
        memcpypgm2ram((void*)argv[0],
          (ROM void*)CONFIG_UPDATE_PAGE, CONFIG_UPDATE_PAGE_LEN);
              break;
```

```
#endif

    default:
                memcpypgm2ram((void*)argv[0],  (ROM  void*)COMMANDS_OK_PAGE,
COMMANDS_OK_PAGE_LEN);
        break;
    }

}
#endif


/************************************************************************
 * Function:        WORD HTTPGetVar(BYTE var, WORD ref, BYTE* val)
 *
 * PreCondition:    None
 *
 * Input:           var       - Variable Identifier
 *                  ref       - Current callback reference with
 *                              respect to 'var' variable.
 *                  val       - Buffer for value storage.
 *
 * Output:          Variable reference as required by application.
 *
 * Side Effects:    None
 *
 * Overview:        This is a callback function from HTTPServer() to
 *                  main application.
 *                  Whenever a variable substitution is required
 *                  on any html pages, HTTPServer calls this function
 *                  8-bit variable identifier, variable reference,
 *                  which indicates whether this is a first call or
 *                  not.  Application should return one character
 *                  at a time as a variable value.
 *
 * Note:            Since this function only allows one character
 *                  to be returned at a time as part of variable
 *                  value, HTTPServer() calls this function
 *                  multiple times until main application indicates
 *                  that there is no more value left for this
 *                  variable.
 *                  On begining, HTTPGetVar() is called with
```

```
*             ref = HTTP_START_OF_VAR to indicate that
*             this is a first call.  Application should
*             use this reference to start the variable value
*             extraction and return updated reference.  If
*             there is no more values left for this variable
*             application should send HTTP_END_OF_VAR.  If
*             there are any bytes to send, application should
*             return other than HTTP_START_OF_VAR and
*             HTTP_END_OF_VAR reference.
*
*             THIS IS AN EXAMPLE CALLBACK.
*             MODIFY THIS AS PER YOUR REQUIREMENTS.
*********************************************************************/
#if defined(STACK_USE_HTTP_SERVER)
WORD HTTPGetVar(BYTE var, WORD ref, BYTE* val)
{
        // Temporary variables designated for storage of a whole return
        // result to simplify logic needed since one byte must be returned
        // at a time.
        static BYTE VarString[25];
#if defined(ENABLE_REMOTE_CONFIG)
        static BYTE VarStringLen;
        BYTE *VarStringPtr;

        BYTE i;
        BYTE *DataSource;
#endif
//putsUART(itoa(var,TEMPString));//coskon
//putrsUART("*");//coskon
         // Identify variable
    switch(var)
    {
    case VAR_IN1:
       *val = IN1_IO ? '0':'1';//IN1
       break;
    case VAR_IN2:
       *val = IN2_IO ? '0':'1';//IN2
       break;
    case VAR_IN3:
       *val = IN3_IO ? '0':'1';//IN3
       break;
                case VAR_IN4:
```

```
    *val = IN4_IO ? '0':'1';//IN4
    break;
  case VAR_INREL1:
    *val = INREL1_IO ? '1':'0'; //Relay1
    break;
  case VAR_INREL2:
    *val = INREL2_IO ? '1':'0'; //relay2
    break;
  case VAR_EXTIN1:
    *val = EXTIN1_IO ? '0':'1';//EXTIN1
    break;
  case VAR_EXTIN2:
    *val = EXTIN2_IO ? '0':'1';//EXTIN2
    break;
  case VAR_EXTIN3:
    *val = EXTIN3_IO ? '0':'1';//EXTIN3
    break;
            case VAR_EXTIN4:
    *val = EXTIN4_IO ? '0':'1';//EXTIN4
    break;
  case VAR_EXTAINV:
    *val = EXTAINV_IO ? '1':'0'; //EXTAINV
    break;
  case VAR_EXTAINI:
    *val = EXTAINI_IO ? '1':'0'; //EXTAINI
    break;
  case VAR_EXTREL1:
    *val = EXTREL1_IO ? '1':'0'; //EXTREL1
    break;
  case VAR_EXTREL2:
    *val = EXTREL2_IO ? '1':'0'; //EXTREL2
    break;
  case VAR_EXTREL3:
//EXTREL3_IO=0;
    *val = EXTREL3_IO ? '1':'0'; //EXTREL3
    break;
            case VAR_EXTREL4:
    *val = EXTREL4_IO ? '1':'0'; //EXTREL4
    break;
  case VAR_EXTREL5:
    *val = EXTREL5_IO ? '1':'0'; //EXTREL5
    break;
```

```
case VAR_EXTREL6:
   *val = EXTREL6_IO ? '1':'0'; //EXTREL6
   break;

case VAR_ANAIN_AN0:
   *val = AN0String[(BYTE)ref];// andriani potensioneter
   if(AN0String[(BYTE)ref] == '\0')
      return HTTP_END_OF_VAR;
            else if(AN0String[(BYTE)++ref] == '\0' )
      return HTTP_END_OF_VAR;
   return ref;
 case VAR_ANAIN_AN1:
   *val = TEMPString[(BYTE)ref];// andriani temperature
   if(TEMPString[(BYTE)ref] == '\0')
      return HTTP_END_OF_VAR;
            else if(TEMPString[(BYTE)++ref] == '\0' )
      return HTTP_END_OF_VAR;
   return ref;
 case VAR_ANAIN_AN14:
   *val = VString[(BYTE)ref];
   if(VString[(BYTE)ref] == '\0')
      return HTTP_END_OF_VAR;
            else if(VString[(BYTE)++ref] == '\0' )
      return HTTP_END_OF_VAR;
   return ref;
 case VAR_ANAIN_AN15:
   *val = LEVString[(BYTE)ref];
   if(LEVString[(BYTE)ref] == '\0')
      return HTTP_END_OF_VAR;
            else if(LEVString[(BYTE)++ref] == '\0' )
      return HTTP_END_OF_VAR;
   return ref;
 case VAR_BUTTON1:
   *val = BUTTON0_IO ? '0':'1';//Button1 on MAXI
   break;
 case VAR_BUTTON2:
   *val = BUTTON1_IO ? '0':'1';//Button2 on MAXI
   break;
 case VAR_DIGIN2:
   *val = EXTIN1_IO ? '1':'0';
   break;
 case VAR_DIGIN3:
```

```
        *val = EXTIN2_IO ? '1':'0';
        break;
          case VAR_STACK_VERSION:
        if(ref == HTTP_START_OF_VAR)
                {
                        strncpypgm2ram((char*)VarString, VERSION, sizeof(VarString));
                }
        *val = VarString[(BYTE)ref];
        if(VarString[(BYTE)ref] == '\0')
          return HTTP_END_OF_VAR;
                else if(VarString[(BYTE)++ref] == '\0' )
          return HTTP_END_OF_VAR;
        return ref;
          case VAR_STACK_DATE:
        if(ref == HTTP_START_OF_VAR)
                {
                        strncpypgm2ram((char*)VarString,  __DATE__  "  "  __TIME__,
 sizeof(VarString));
                }
       *val = VarString[(BYTE)ref];
       if(VarString[(BYTE)ref] == '\0')
          return HTTP_END_OF_VAR;
                else if(VarString[(BYTE)++ref] == '\0' )
          return HTTP_END_OF_VAR;
      return ref;

#if defined(ENABLE_REMOTE_CONFIG)
   case VAR_MAC_ADDRESS:
     if ( ref == HTTP_START_OF_VAR )
     {
       VarStringLen = 2*6+5;        // 17 bytes: 2 for each of the 6 address bytes + 5
octet spacers

          // Format the entire string
       i = 0;
       VarStringPtr = VarString;
       while(1)
       {
            *VarStringPtr++ = btohexa_high(AppConfig.MyMACAddr.v[i]);
            *VarStringPtr++ = btohexa_low(AppConfig.MyMACAddr.v[i]);
            if(++i == 6)
                  break;
```

```
            *VarStringPtr++ = '-';
        }
    }

            // Send one byte back to the calling function (the HTTP Server)
            *val = VarString[(BYTE)ref];

    if ( (BYTE)++ref == VarStringLen )
        return HTTP_END_OF_VAR;

    return ref;

case VAR_IP_ADDRESS:
case VAR_SUBNET_MASK:
case VAR_GATEWAY_ADDRESS:
        // Check if ref == 0 meaning that the first character of this
        // variable needs to be returned
    if ( ref == HTTP_START_OF_VAR )
    {
            // Decide which 4 variable bytes to send back
            if(var == VAR_IP_ADDRESS)
                    DataSource = (BYTE*)&AppConfig.MyIPAddr;
                else if(var == VAR_SUBNET_MASK)
                    DataSource = (BYTE*)&AppConfig.MyMask;
                else if(var == VAR_GATEWAY_ADDRESS)
                    DataSource = (BYTE*)&AppConfig.MyGateway;

            // Format the entire string
            VarStringPtr = VarString;
            i = 0;
            while(1)
            {
                    uitoa((WORD)*DataSource++, VarStringPtr);
                    VarStringPtr += strlen(VarStringPtr);
                    if(++i == 4)
                        break;
                    *VarStringPtr++ = '.';
                }
                VarStringLen = strlen(VarString);
        }

                // Send one byte back to the calling function (the HTTP Server)
```

```
                *val = VarString[(BYTE)ref];

                // If this is the last byte to be returned, return
                // HTTP_END_OF_VAR so the HTTP server won't keep calling this
                // application callback function
        if ( (BYTE)++ref == VarStringLen )
          return HTTP_END_OF_VAR;

        return ref;

      case VAR_DHCP_TRUE:
      case VAR_DHCP_FALSE:
          // Check if ref == 0 meaning that the first character of this
          // variable needs to be returned
        if ( ref == HTTP_START_OF_VAR )
        {
              if((var == VAR_DHCP_TRUE) ^ AppConfig.Flags.bIsDHCPEnabled)
                return HTTP_END_OF_VAR;

          VarStringLen = 7;
                      memcpypgm2ram(VarString, (rom void *)"checked", 7);
       }

                *val = VarString[(BYTE)ref];

       if ( (BYTE)++ref == VarStringLen )
          return HTTP_END_OF_VAR;

       return ref;
 #endif
    }

   return HTTP_END_OF_VAR;
}
#endif


#if     defined(STACK_USE_FTP_SERVER)     &&     defined(MPFS_USE_EEPROM)     &&
defined(STACK_USE_MPFS)
ROM char FTP_USER_NAME[]    = "admin";
ROM char FTP_USER_PASS[]    = "embeddedview";
#undef FTP_USER_NAME_LEN
```

```
#define FTP_USER_NAME_LEN   (sizeof(FTP_USER_NAME)-1)
#define FTP_USER_PASS_LEN   (sizeof(FTP_USER_PASS)-1)
//coskon
BOOL FTPVerify(BYTE *login, BYTE *password)
{
int i;
   if ( !memcmppgm2ram(login, (ROM void*)FTP_USER_NAME, FTP_USER_NAME_LEN) )
   {
                  for(i=0;i<strlen(AppConfig.FtpPassword);i++){
       if(password[i]!=AppConfig.FtpPassword[i])
                                 return FALSE;
                        }
    return TRUE;
    }
 return FALSE;
 }
 #endif



 /*****************************************************************************
  * Function:       void InitializeBoard(void)
  *
  * PreCondition:    None
  *
  * Input:          None
  *
  * Output:         None
  *
  * Side Effects:    None
  *
  * Overview:       Initialize board specific hardware.
  *
  * Note:           None
  *****************************************************************************/
 static void InitializeBoard(void)
 {
        // LEDs
        LED0_TRIS = 0;
        LED1_TRIS = 0;
        LED2_TRIS = 0;
```

```
        LCDLIGHT_TRIS = 0;

        INREL1_TRIS = 0;
        INREL2_TRIS = 0;
        LED0_IO = 0;
        LED1_IO = 0;
        LED2_IO = 0;
        LCDLIGHT_IO = 1;
        INREL1_IO = 0;
        INREL2_IO = 0;


  #ifdef __C30__
        #if defined(__dsPIC33F__) || defined(__PIC24H__)
        // Crank up the core frequency
        PLLFBD = 38;                        // Multiply by 40 for 160MHz VCO output
  (8MHz XT oscillator)
        CLKDIV = 0x0000;                    // FRC: divide by 2, PLLPOST: divide by 2,
  PLLPRE: divide by 2

        // Port I/O
        AD1PCFGHbits.PCFG23 = 1;    // Make RA7 (BUTTON1) a digital input
        #endif

        // ADC
        AD1CON1 = 0x84E4;                   // Turn on, auto sample start, auto-
  convert, 12 bit mode (on parts with a 12bit A/D)
        AD1CON2 = 0x0404;                   // AVdd, AVss, int every 2 conversions,
  MUXA only, scan
        AD1CON3 = 0x1003;                   // 16 Tad auto-sample, Tad = 3*Tcy
    AD1CHS = 0;                                 // Input to AN0 (potentiometer)
        AD1PCFGbits.PCFG5 = 0;              // Disable digital input on AN5
  (potentiometer)
        AD1PCFGbits.PCFG4 = 0;              // Disable digital input on AN4 (TC1047A
  temp sensor)
        AD1CSSL = 1<<5;                         // Scan pot

//      // Enable ADC interrupt
//      IFS0bits.AD1IF = 0;
//      IEC0bits.AD1IE = 1;
```

```
        // UART
        UARTTX_TRIS = 0;
        UARTRX_TRIS = 1;
        UBRG = (INSTR_FREQ+8ul*BAUD_RATE)/16/BAUD_RATE-1;
        UMODE = 0x8000;             // UARTEN set
        USTA = 0x0400;              // UTXEN set
#else
        // Enable 4x/5x PLL on PIC18F87J10, PIC18F97J60, etc.
    OSCTUNE = 0x40;

        // Set up analog features of PORTA

        // PICDEM.net 2 board has POT on AN2, Temp Sensor on AN3
        #if defined(PICDEMNET2) || defined(OLIMEX_MAXI)
                ADCON0 = 0x09;              // ADON, Channel 2
                ADCON1 = 0x0B;              // Vdd/Vss is +/-REF, AN0, AN1, AN2, AN3
are analog
            TRISA = 0x2F;
        #elif defined(__18F452)
                ADCON0 = 0x81;              // ADON, Channel 0, Fosc/32
                ADCON1 = 0x8E;              // Right justified, Fosc/32, AN0 only anlog,
VREF+/VREF- are VDD/VSS
            TRISA = 0x23;
        #elif defined(PICDEMZ)
                ADCON0 = 0x81;              // ADON, Channel 0, Fosc/32
                ADCON1 = 0x0F;              // Vdd/Vss is +/-REF, AN0, AN1, AN2, AN3
are all digital
        #else
                ADCON0 = 0x01;              // ADON, Channel 0
                ADCON1 = 0x0E;              // Vdd/Vss is +/-REF, AN0 is analog
            TRISA = 0x23;
        #endif
        ADCON2 = 0xBE;                      // Right justify, 20TAD ACQ time, Fosc/64
(~21.0kHz)


    // Enable internal PORTB pull-ups
    INTCON2bits.RBPU = 0;

        // Configure USART
    TXSTA = 0x20;
 #if defined(FS_USB)
```

```
    RCSTA = 0x80;               // PICDEM FS USB demo board has UART RX pin multipled
with SPI, we must not enable the UART RX functionality
#else
    RCSTA = 0x90;
#endif

        // See if we can use the high baud rate setting
#if ((INSTR_FREQ+2*BAUD_RATE)/BAUD_RATE/4 - 1) <= 255
        SPBRG = (INSTR_FREQ+2*BAUD_RATE)/BAUD_RATE/4 - 1;
        TXSTAbits.BRGH = 1;
#else   // Use the low baud rate setting
        SPBRG = (INSTR_FREQ+8*BAUD_RATE)/BAUD_RATE/16 - 1;
#endif


        // Enable Interrupts
        RCONbits.IPEN = 1;          // Enable interrupt priorities
    INTCONbits.GIEH = 1;
    INTCONbits.GIEL = 1;

    // Do a calibration A/D conversion
        #if  defined(__18F87J10)  ||  defined(__18F86J15)  ||  defined(__18F86J10)  ||
defined(__18F85J15)     ||     defined(__18F85J10)     ||     defined(__18F67J10)     ||
defined(__18F66J15)     ||     defined(__18F66J10)     ||     defined(__18F65J15)     ||
defined(__18F65J10)     ||     defined(__18F97J60)     ||     defined(__18F96J65)     ||
defined(__18F96J60)     ||     defined(__18F87J60)     ||     defined(__18F86J65)     ||
defined(__18F86J60)     ||     defined(__18F67J60)     ||     defined(__18F66J65)     ||
defined(__18F66J60) || \
        defined(_18F87J10)     ||  defined(_18F86J15)     ||  defined(_18F86J10)     ||
defined(_18F85J15)      ||     defined(_18F85J10)      ||     defined(_18F67J10)      ||
defined(_18F66J15)      ||     defined(_18F66J10)      ||     defined(_18F65J15)      ||
defined(_18F65J10)      ||     defined(_18F97J60)      ||     defined(_18F96J65)      ||
defined(_18F96J60)      ||     defined(_18F87J60)      ||     defined(_18F86J65)      ||
defined(_18F86J60)      ||     defined(_18F67J60)      ||     defined(_18F66J65)      ||
defined(_18F66J60)
            ADCON0bits.ADCAL = 1;
          ADCON0bits.GO = 1;
            while(ADCON0bits.GO);
            ADCON0bits.ADCAL = 0;
        #endif

//      // Enable ADC interrupt
```

```
//      PIR1bits.ADIF = 0;
//      PIE1bits.ADIE = 1;

#endif

#if defined(DSPICDEM11)
        // Deselect the LCD controller (PIC18F252 onboard) to ensure there is no SPI2
contention
        LCDCTRL_CS_TRIS = 0;
        LCDCTRL_CS_IO = 1;

        // Hold the codec in reset to ensure there is no SPI2 contention
        CODEC_RST_TRIS = 0;
        CODEC_RST_IO = 0;
#elif defined(DSPICDEMNET1) || defined(DSPICDEMNET2)
        // Hold Si3021 in reset to keep the speaker as quiet as possible
        TRISGbits.TRISG8 = 0;
        LATGbits.LATG8 = 0;

        // Ensure that the SRAM does not interfere with RTL8019AS communications
        SRAM_CE_ADPCFG = 1;
        SRAM_OE_ADPCFG = 1;
        SRAM_CE_IO = 1;
        SRAM_OE_IO = 1;
        SRAM_CE_TRIS = 0;
        SRAM_OE_TRIS = 0;
#elif defined(PIC18F67J60_TEST_BOARD)
        // Make PIC SPI1 clock (SCK) not cause contention with U2:D level shifter. U2:D
drives RC3 as well, by default on board revision 1.
        TRISEbits.TRISE1 = 0;
        LATEbits.LATE1 = 1;
#endif

}

/*****************************************************************************
 * Function:        void InitAppConfig(void)
 *
 * PreCondition:    MPFSInit() is already called.
 *
 * Input:           None
 *
```

```
 * Output:        Write/Read non-volatile config variables.
 *
 * Side Effects:  None
 *
 * Overview:      None
 *
 * Note:          None
 ********************************************************************/
static void InitAppConfig(void)
{
#if defined(MPFS_USE_EEPROM) && defined(STACK_USE_MPFS)
    BYTE c;
    BYTE *p;
#endif

        AppConfig.Flags.bIsDHCPEnabled = TRUE;
        AppConfig.Flags.bInConfigMode = TRUE;
        AppConfig.MyMACAddr.v[0] = MY_DEFAULT_MAC_BYTE1;
        AppConfig.MyMACAddr.v[1] = MY_DEFAULT_MAC_BYTE2;
        AppConfig.MyMACAddr.v[2] = MY_DEFAULT_MAC_BYTE3;
        AppConfig.MyMACAddr.v[3] = MY_DEFAULT_MAC_BYTE4;
        AppConfig.MyMACAddr.v[4] = MY_DEFAULT_MAC_BYTE5;
        AppConfig.MyMACAddr.v[5] = MY_DEFAULT_MAC_BYTE6;
        AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 |
MY_DEFAULT_IP_ADDR_BYTE2<<8ul | MY_DEFAULT_IP_ADDR_BYTE3<<16ul |
MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
        AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
        AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 |
MY_DEFAULT_MASK_BYTE2<<8ul | MY_DEFAULT_MASK_BYTE3<<16ul |
MY_DEFAULT_MASK_BYTE4<<24ul;
        AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
        AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 |
MY_DEFAULT_GATE_BYTE2<<8ul | MY_DEFAULT_GATE_BYTE3<<16ul |
MY_DEFAULT_GATE_BYTE4<<24ul;
        AppConfig.PrimaryDNSServer.Val = MY_DEFAULT_PRIMARY_DNS_BYTE1 |
MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul | MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul
| MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
        AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1 |
MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul |
MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul |
MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;
```

```
        // Load the default NetBIOS Host Name
        memcpypgm2ram(AppConfig.NetBIOSName,                    (ROM
void*)MY_DEFAULT_HOST_NAME, 16);
        FormatNetBIOSName(AppConfig.NetBIOSName);

#if defined(MPFS_USE_EEPROM) && defined(STACK_USE_MPFS)
   p = (BYTE*)&AppConfig;


   XEEBeginRead(0x0000);
   c = XEERead();
   XEEEndRead();

   // When a record is saved, first byte is written as 0x57 to indicate
   // that a valid record was saved.  Note that older stack versions
        // used 0x55.  This change has been made to so old EEPROM contents
        // will get overwritten.  The AppConfig() structure has been changed,
        // resulting in parameter misalignment if still using old EEPROM
        // contents.
   if(c == 0x60u)
   {
      XEEBeginRead(0x0001);
      for ( c = 0; c < sizeof(AppConfig); c++ )
         *p++ = XEERead();
      XEEEndRead();
   }
   else
      SaveAppConfig();
#endif
}

#if defined(MPFS_USE_EEPROM) && defined (STACK_USE_MPFS)
static void SaveAppConfig(void)
{
   BYTE c;
   BYTE *p;

   p = (BYTE*)&AppConfig;
   XEEBeginWrite(0x0000);
   XEEWrite(0x60);
   for ( c = 0; c < sizeof(AppConfig); c++ )
   {
```

```
        XEEWrite(*p++);
    }

    XEEEndWrite();
}
#endif



#if defined(STACK_USE_UART)
#define MAX_USER_RESPONSE_LEN  (20u)
static void SetConfig(void)
{
    BYTE response[MAX_USER_RESPONSE_LEN];
    IP_ADDR tempIPValue;
    IP_ADDR *destIPValue;
        WORD_VAL wvTemp;
    BOOL bQuit = FALSE;
        int i;

  unsigned char tempres;

        while(!bQuit)
        {
                // Display the menu
          putrsUART("\r\n\r\n\rEmbeddedView TCP/IP Config Application ("VERSION", "
__DATE__ ")\r\n\r\n");
          putrsUART("\t1: Change serial number:\t\t");
                wvTemp.v[1] = AppConfig.MyMACAddr.v[4];
                wvTemp.v[0] = AppConfig.MyMACAddr.v[5];
                uitoa(wvTemp.Val, response);
                putsUART(response);
                putrsUART("\r\n\t2: Change host name:\t\t\t");
                putsUART(AppConfig.NetBIOSName);
         putrsUART("\r\n\t3: Change static IP address:\t\t");
         DisplayIPValue(AppConfig.MyIPAddr);
         putrsUART("\r\n\t4: Change static gateway address:\t");
         DisplayIPValue(AppConfig.MyGateway);
         putrsUART("\r\n\t5: Change static subnet mask:\t\t");
         DisplayIPValue(AppConfig.MyMask);
              putrsUART("\r\n\t6: Change static primary DNS server:\t");
        DisplayIPValue(AppConfig.PrimaryDNSServer);
```

```
            putrsUART("\r\n\t7: Change static secondary DNS server:\t");
        DisplayIPValue(AppConfig.SecondaryDNSServer);
        putrsUART("\r\n\t8: ");
            putrsUART((ROM   BYTE*)(AppConfig.Flags.bIsDHCPEnabled   ?   "Dis"   :
"En"));
            putrsUART("able DHCP & IP Gleaning:\t\tDHCP is currently ");
            putrsUART((ROM  BYTE*)(AppConfig.Flags.bIsDHCPEnabled  ?  "enabled"  :
"disabled"));
        putrsUART("\r\n\t9: Download MPFS image.");
//coskon
//              putrsUART("\r\n\tA: Change FtpPassword:\t\t\t");
//              putrsUART("*******");

        putrsUART("\r\n\t0: Save & Quit.");
        putrsUART("\r\nEnter a menu choice: ");


            // Wait for the user to press a key
        while(!DataRdyUART());

            putrsUART((ROM char*)"\r\n");

            // Execute the user selection
        switch(ReadUART())
        {
                        case 'T':
                putrsUART("Put extension headers and press any key");
                while(!DataRdyUART());
                putrsUART("\r\n");
                tempres = TestExt();
                if (tempres == TEST_OK)
                {
                        putrsUART("EXT test OK, testing X-tension\r\n");
                        tempres = TestXPort();
                   }
                switch(tempres)
                {
                        case TEST_OK:
                                putrsUART("Extensions OK, Check Inputs\r\n");
                                LED0_TRIS = 0;
                                LED0_IO = 1;
```

```
                //InitRTC();
                // test EXT OK, leave the RTC to blink LED0 and
test optrons

                TestOptrons();
                while (1);
                break;
        case TEST_GND:
                putrsUART("Pin to GND!!!!!!!!!");
                break;
        case TEST_VCC:
                putrsUART("Pin to VCC!!!!!!!!!");
                break;
        case TEST_ZERO_A:
                putrsUART("Short on PORT A!!!!!!!");
                break;
                        case TEST_ZERO_B:
                putrsUART("Short on PORT B!!!!!!!");
                break;
        case TEST_ZERO_C:
                putrsUART("Short on PORT C!!!!!!!");
                break;
        case TEST_ZERO_D:
                putrsUART("Short on PORT D!!!!!!!");
                break;
        case TEST_ZERO_E:
                putrsUART("Short on PORT E!!!!!!!");
                break;
                        case TEST_ZERO_F:
                putrsUART("Short on PORT F!!!!!!!");
                break;
                        case TEST_ZERO_G:
                putrsUART("Short on PORT G!!!!!!!");
                break;
                        case TEST_ZERO_H:
                putrsUART("Short on PORT H!!!!!!!");
                break;
                        case TEST_ZERO_J:
                putrsUART("Short on PORT J!!!!!!!");
                break;
        case TEST_RTC:
                putrsUART("RTC fault");
                break;
```

```
                        }
                        // test fault - do nothing
                        LED0_TRIS = 0;
                        LED0_IO = 0;
                        while(1);
                        break;
                case '1':
                        putrsUART("New setting: ");
                        if(ReadStringUART(response, sizeof(response)))
                        {
                                wvTemp.Val = atoi((char*)response);
                            AppConfig.MyMACAddr.v[4] = wvTemp.v[1];
                          AppConfig.MyMACAddr.v[5] = wvTemp.v[0];
                        }
                    break;

                    case '2':
                            putrsUART("New setting: ");
                   ReadStringUART(response,          sizeof(response)          >
sizeof(AppConfig.NetBIOSName) ? sizeof(AppConfig.NetBIOSName) : sizeof(response));
                            if(response[0] != '\0')
                            {
                                    memcpy(AppConfig.NetBIOSName,
(void*)response, sizeof(AppConfig.NetBIOSName));
                          FormatNetBIOSName(AppConfig.NetBIOSName);
                            }
                            break;

                case '3':
                   destIPValue = &AppConfig.MyIPAddr;
                   goto ReadIPConfig;

                case '4':
                   destIPValue = &AppConfig.MyGateway;
                   goto ReadIPConfig;

                case '5':
                   destIPValue = &AppConfig.MyMask;
                   goto ReadIPConfig;

                case '6':
                   destIPValue = &AppConfig.PrimaryDNSServer;
```

```
                goto ReadIPConfig;

                  case '7':
                destIPValue = &AppConfig.SecondaryDNSServer;
                goto ReadIPConfig;

ReadIPConfig:
                        putrsUART("New setting: ");
                ReadStringUART(response, sizeof(response));

                if(StringToIPAddress(response, &tempIPValue))
                  destIPValue->Val = tempIPValue.Val;
                        else
                  putrsUART("Invalid input.\r\n");

                break;


              case '8':
                AppConfig.Flags.bIsDHCPEnabled                    =
!AppConfig.Flags.bIsDHCPEnabled;
                break;

              case '9':
                        #if          defined(MPFS_USE_EEPROM)          &&
defined(STACK_USE_MPFS)
                DownloadMPFS();
                        #endif
              break;
//              case 'A':
//coskon
//                        putrsUART("Enter old password:");
//              ReadStringUART(response,          sizeof(response)          >
sizeof(AppConfig.FtpPassword) ? sizeof(AppConfig.FtpPassword) : sizeof(response));
//                        i=0;
//                        while(response[i]!='\0' && i<20){
//              if(response[i]!=AppConfig.FtpPassword[i])
//                              break;
//                              i++;}
//                      putrsUART("\nEnter new password: ");
//              ReadStringUART(response,          sizeof(response)          >
sizeof(AppConfig.FtpPassword) ? sizeof(AppConfig.FtpPassword) : sizeof(response));
```

```
//                          if(response[0] != '\0')
//                          {
//                                  memcpy(AppConfig.FtpPassword, (void*)response,
sizeof(AppConfig.FtpPassword));
//                          }
//                          break;
//
                case '0':
                        bQuit = TRUE;
                            #if         defined(MPFS_USE_EEPROM)          &&
defined(STACK_USE_MPFS)
                        SaveAppConfig();
                                putrsUART("Settings saved.\r\n");
                            #else
                                putrsUART("External MPFS not enabled -- settings
will be lost at reset.\r\n");
                            #endif
                    break;
                }
        }
}
#endif //#if defined(STACK_USE_UART)


#if     defined(MPFS_USE_EEPROM)     &&     defined(STACK_USE_MPFS)     &&
defined(STACK_USE_UART)
/*****************************************************************************
 * Function:        BOOL DownloadMPFS(void)
 *
 * PreCondition:    MPFSInit() is already called.
 *
 * Input:           None
 *
 * Output:          TRUE if successful
 *                  FALSE otherwise
 *
 * Side Effects:    This function uses 128 bytes of Bank 4 using
 *                  indirect pointer.  This requires that no part of
 *                  code is using this block during or before calling
 *                  this function.  Once this function is done,
 *                  that block of memory is available for general use.
 *
```

```
  * Overview:       This function implements XMODEM protocol to
  *                 be able to receive a binary file from PC
  *                 applications such as HyperTerminal.
  *
  * Note:           In current version, this function does not
  *                 implement user interface to set IP address and
  *                 other informations.  User should create their
  *                 own interface to allow user to modify IP
  *                 information.
  *                 Also, this version implements simple user
  *                 action to start file transfer.  User may
  *                 evaulate its own requirement and implement
  *                 appropriate start action.
  *
  ********************************************************************/
#define XMODEM_SOH     0x01u
#define XMODEM_EOT     0x04u
#define XMODEM_ACK     0x06u
#define XMODEM_NAK     0x15u
#define XMODEM_CAN     0x18u
#define XMODEM_BLOCK_LEN 128u

static BOOL DownloadMPFS(void)
{
  enum SM_MPFS
  {
    SM_MPFS_SOH,
    SM_MPFS_BLOCK,
    SM_MPFS_BLOCK_CMP,
    SM_MPFS_DATA,
  } state;

  BYTE c;
  MPFS handle;
  BOOL lbDone;
  BYTE blockLen;
  BYTE lResult;
  BYTE tempData[XMODEM_BLOCK_LEN];
  TICK lastTick;
  TICK currentTick;

state = SM_MPFS_SOH;
```

```
lbDone = FALSE;

handle = MPFSFormat();

// Notify the host that we are ready to receive...
lastTick = TickGet();
do
{
   currentTick = TickGet();
   if ( TickGetDiff(currentTick, lastTick) >= (TICK_SECOND/2) )
   {
      lastTick = TickGet();
                  while(BusyUART());
      WriteUART(XMODEM_NAK);

      /*
       * Blink LED to indicate that we are waiting for
       * host to send the file.
       */
       #if defined(OLIMEX_HW)
       LED0_IO ^= 1;
       #else
       INREL2_IO ^= 1;
       #endif
    }

  } while(!DataRdyUART());


  while(!lbDone)
  {
     if(DataRdyUART())
     {
        // Toggle LED as we receive the data from host.
        #if defined(OLIMEX_HW)
        LED0_IO ^= 1;
        #else
        INREL2_IO ^= 1;
        #endif
        c = ReadUART();
     }
     else
```

```
    {
      // Real application should put some timeout to make sure
      // that we do not wait forever.
      continue;
    }

    switch(state)
    {
    default:
      if ( c == XMODEM_SOH )
      {
        state = SM_MPFS_BLOCK;
      }
      else if ( c == XMODEM_EOT )
      {
        // Turn off LED when we are done.
        #if defined(OLIMEX_HW)
        LED0_IO = 1;
        #else
        INREL2_IO = 1;
        #endif

        MPFSClose();
                              while(BusyUART());
        WriteUART(XMODEM_ACK);
        lbDone = TRUE;
    }
    else
                    {
                              while(BusyUART());
                              WriteUART(XMODEM_NAK);
                    }

    break;

case SM_MPFS_BLOCK:

    // We do not use block information.
    lResult = XMODEM_ACK;
    blockLen = 0;
    state = SM_MPFS_BLOCK_CMP;
    break;
```

```
    case SM_MPFS_BLOCK_CMP:

        // We do not use 1's comp. block value.
        state = SM_MPFS_DATA;
        break;

    case SM_MPFS_DATA:

        // Buffer block data until it is over.
        tempData[blockLen++] = c;
        if ( blockLen > XMODEM_BLOCK_LEN )
        {

            // We have one block data. Write it to EEPROM.
            MPFSPutBegin(handle);

            lResult = XMODEM_ACK;
            for ( c = 0; c < XMODEM_BLOCK_LEN; c++ )
                MPFSPut(tempData[c]);

            handle = MPFSPutEnd();

                                    while(BusyUART());
            WriteUART(lResult);
            state = SM_MPFS_SOH;
          }
          break;

      }

    }

    return TRUE;
}
#endif  // #if defined(MPFS_USE_EEPROM) && defined(STACK_USE_MPFS)

// NOTE: Name[] must be at least 16 characters long.
// It should be exactly 16 characters, as defined by NetBIOS spec.
static void FormatNetBIOSName(BYTE Name[])
{
        BYTE i;
```

```
        Name[15] = '\0';
        strupr((char*)Name);
        i = 0;
        while(i < 15u)
        {
                if(Name[i] == '\0')
                {
                        while(i < 15u)
                        {
                                Name[i++] = ' ';
                        }
                        break;
                }
                i++;
        }
}


#if defined(STACK_USE_SNMP_SERVER)

/*
 * Trap information.
 * This table maintains list of intereseted receivers
 * who should receive notifications when some interesting
 * event occurs.
 */
#define TRAP_TABLE_SIZE        (2)
#define MAX_COMMUNITY_LEN      (8)
typedef struct _TRAP_INFO
{
  BYTE Size;
  struct
  {
    BYTE communityLen;
    char community[MAX_COMMUNITY_LEN];
    IP_ADDR IPAddress;
    struct
    {
      unsigned int bEnabled : 1;
    } Flags;
```

```
  } table[TRAP_TABLE_SIZE];
} TRAP_INFO;

/*
 * Initialize trap table with no entries.
 */
TRAP_INFO trapInfo = { TRAP_TABLE_SIZE };


BOOL SNMPValidate(SNMP_ACTION SNMPAction, char* community)
{
   return TRUE;
}



// Only dynamic variables with ASCII_STRING or OCTET_STRING data type
// needs to be handled.
BOOL SNMPIsValidSetLen(SNMP_ID var, BYTE len)
{
   switch(var)
   {
   case TRAP_COMMUNITY:
      if ( len < MAX_COMMUNITY_LEN+1 )
         return TRUE;
      break;

#if defined(USE_LCD)
    case LCD_DISPLAY:
      if ( len < sizeof(LCDText)+1 )
         return TRUE;
      break;
#endif
    }
    return FALSE;
}


// Only dynamic read-write variables needs to be handled.
BOOL SNMPSetVar(SNMP_ID var, SNMP_INDEX index, BYTE ref, SNMP_VAL val)
{
   switch(var)
```

```
    {
    case LED_D5:
        LED1_IO = val.byte;
        return TRUE;

    case LED_D6:
        LED2_IO = val.byte;
        return TRUE;

    case TRAP_RECEIVER_IP:
        // Make sure that index is within our range.
        if ( index < trapInfo.Size )
        {
            // This is just an update to an existing entry.
            trapInfo.table[index].IPAddress.Val = val.dword;
            return TRUE;
        }
        else if ( index < TRAP_TABLE_SIZE )
        {
            // This is an addition to table.
            trapInfo.table[index].IPAddress.Val = val.dword;
            trapInfo.table[index].communityLen = 0;
            trapInfo.Size++;
            return TRUE;
        }
        break;

case TRAP_RECEIVER_ENABLED:
    // Make sure that index is within our range.
    if ( index < trapInfo.Size )
    {
        // Value of '1' means Enabled".
        if ( val.byte == 1 )
            trapInfo.table[index].Flags.bEnabled = 1;
        // Value of '0' means "Disabled.
        else if ( val.byte == 0 )
            trapInfo.table[index].Flags.bEnabled = 0;
        else
            // This is unknown value.
            return FALSE;
        return TRUE;
    }
```

```
    // Given index is more than our current table size.
    // If it is within our range, treat it as an addition to table.
    else if ( index < TRAP_TABLE_SIZE )
    {
      // Treat this as an addition to table.
      trapInfo.Size++;
      trapInfo.table[index].communityLen = 0;
    }

    break;

case TRAP_COMMUNITY:
    // Since this is a ASCII_STRING data type, SNMP will call with
    // SNMP_END_OF_VAR to indicate no more bytes.
    // Use this information to determine if we just added new row
    // or updated an existing one.
    if ( ref ==  SNMP_END_OF_VAR )
    {
      // Index equal to table size means that we have new row.
      if ( index == trapInfo.Size )
        trapInfo.Size++;

      // Length of string is one more than index.
      trapInfo.table[index].communityLen++;

      return TRUE;
    }

    // Make sure that index is within our range.
    if ( index < trapInfo.Size )
    {
      // Copy given value into local buffer.
      trapInfo.table[index].community[ref] = val.byte;
      // Keep track of length too.
      // This may not be NULL terminate string.
      trapInfo.table[index].communityLen = (BYTE)ref;
      return TRUE;
    }
    break;

#if defined(USE_LCD)
  case LCD_DISPLAY:
```

```
        // Copy all bytes until all bytes are transferred
        if ( ref != SNMP_END_OF_VAR )
        {
           LCDText[ref] = val.byte;
           LCDText[ref+1] = 0;
        }
        else
        {
                        LCDUpdate();
        }

      return TRUE;
#endif

   }

   return FALSE;
}

// Only sequence index needs to be handled in this function.
BOOL SNMPGetNextIndex(SNMP_ID var, SNMP_INDEX *index)
{
  SNMP_INDEX tempIndex;

  tempIndex = *index;

  switch(var)
  {
  case TRAP_RECEIVER_ID:
    // There is no next possible index if table itself is empty.
    if ( trapInfo.Size == 0 )
       return FALSE;

    // INDEX_INVALID means start with first index.
    if ( tempIndex == SNMP_INDEX_INVALID )
    {
       *index = 0;
       return TRUE;
    }
    else if ( tempIndex < (trapInfo.Size-1) )
    {
      *index = tempIndex+1;
```

```
            return TRUE;
        }
        break;
    }
    return FALSE;
}
BOOL SNMPGetVar(SNMP_ID var, SNMP_INDEX index, BYTE *ref, SNMP_VAL* val)
{
    BYTE myRef;

    myRef = *ref;

    switch(var)
    {
    case SYS_UP_TIME:
        val->dword = TickGet();
        return TRUE;

    case LED_D5:
        val->byte = LED1_IO;
        return TRUE;

    case LED_D6:
        val->byte = LED2_IO;
        return TRUE;

    case PUSH_BUTTON:
        // There is only one button - meaning only index of 0 is allowed.
        val->byte = BUTTON0_IO;
        return TRUE;

    case ANALOG_POT0:
        val->word = atoi((char*)AN0String);
        return TRUE;

    case TRAP_RECEIVER_ID:
        if ( index < trapInfo.Size )
        {
            val->byte = index;
            return TRUE;
        }
        break;
```

```
    case TRAP_RECEIVER_ENABLED:
      if ( index < trapInfo.Size )
      {
        val->byte = trapInfo.table[index].Flags.bEnabled;
        return TRUE;
      }
      break;

    case TRAP_RECEIVER_IP:
      if ( index < trapInfo.Size )
      {
        val->dword = trapInfo.table[index].IPAddress.Val;
        return TRUE;
      }
      break;

    case TRAP_COMMUNITY:
      if ( index < trapInfo.Size )
      {
        if ( trapInfo.table[index].communityLen == 0 )
          *ref = SNMP_END_OF_VAR;
        else
        {
          val->byte = trapInfo.table[index].community[myRef];

          myRef++;

          if ( myRef == trapInfo.table[index].communityLen )
            *ref = SNMP_END_OF_VAR;
          else
            *ref = myRef;
        }
        return TRUE;
      }
      break;

#if defined(USE_LCD)
    case LCD_DISPLAY:
      if ( LCDText[0] == 0 )
        myRef = SNMP_END_OF_VAR;
      else
```

```
    {
        val->byte = LCDText[myRef++];
        if ( LCDText[myRef] == 0 )
            myRef = SNMP_END_OF_VAR;
    }

    *ref = myRef;
    return TRUE;
#endif
  }

  return FALSE;
}
#endif  //#if defined(STACK_USE_SNMP_SERVER)
```

ΠΑΡΑΡΤΗΜΑ Β

Κώδικας (ιστοσελίδας)

Ο κώδικας είναι γραμμένος σε HDML και είναι ο εξής:

```
<html>
	<head>
		<title>EmbeddedView TCP/IP Stack Home</title>
		<script language="JavaScript">
			var	xmlHttp;
			var ObjArray = new Array;

			function GetXmlHttpObject(handler)
			{
				var objXmlHttp = null;

				if(navigator.userAgent.indexOf("MSIE")>=0)
				{
					var ClassName = "Msxml2.XMLHTTP";
					if(navigator.appVersion.indexOf("MSIE 5.5")>=0)
					{
					ClassName = "Microsoft.XMLHTTP";
					}

					try
					{
						objXmlHttp = new
ActiveXObject(ClassName);
						objXmlHttp.onreadystatechange = handler;
						return objXmlHttp;
					}
					catch(e)
					{
						alert("Error: ActiveX scripting may be
disabled.");
						return;
					}
				}
				else
```

```
                {
                        try
                        {
                                objXmlHttp = new XMLHttpRequest();
                                objXmlHttp.onload = handler;
                                objXmlHttp.onerror = handler;
                                return objXmlHttp;
                        }
                        catch(e)
                        {
                                alert("Error: Browser may not be supported
or browser security restrictions are too high.  XMLHttpRequest() support is required.");
                        }
                }
        }

        function StateChanged()
        {
                if(xmlHttp.readyState == 4 || xmlHttp.readyState ==
"complete")
                {

        document.getElementById("txtAutoUpdateStatus").innerHTML=xmlHttp.responseT
ext;
                        xmlHttp = null;
                        UpdateStatus();
                }
        }


        function UpdateStatus()
        {
                xmlHttp = GetXmlHttpObject(StateChanged);
                xmlHttp.open("GET", "Status.cgi" , true);
                xmlHttp.send(null);
        }

        function GetServerFile(FileName, AssignTo)
        {
                var NiftyObj = new Object();
                NiftyObj.XMLDevice = new
  GetXmlHttpObject(StateChanged2);
```

```
                NiftyObj.XMLDevice.open("GET", FileName, true);
                NiftyObj.XMLDevice.send(null);
                NiftyObj.Text = AssignTo;
                ObjArray.push(NiftyObj);
        }

        function StateChanged2()
        {
                for(i in ObjArray)
                {
                        if(ObjArray[i].XMLDevice.readyState == 4 ||
ObjArray[i].XMLDevice.readyState == "complete")
                        {
                                if(ObjArray[i].Text != "")
                                {

        document.getElementById(ObjArray[i].Text).innerHTML=ObjArray[i].XMLDevice.re
sponseText;
                                }

                                if(ObjArray[i].Text ==
"txtAutoUpdateStatus")
                                {
                                        GetServerFile("Status.cgi",
"txtAutoUpdateStatus");
                                }

                                delete ObjArray[i].XMLDevice;
                                delete ObjArray[i];
                        }
                }
        }



        </script>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-7">
    </head>
```

```
<body bgcolor="white" onLoad="UpdateStatus();
GetServerFile('Version.cgi','txtStackVersion');GetServerFile('IP.cgi','txtIP');
GetServerFile('BuildDate.cgi','txtBuildDate'); ">
<table border="0" width="675" height="138">
<tr>
<td width="19%" height="134"><b>
<font size="6" face="MS Sans Serif"><img src="TEI_Peir.gif"
width="120" height="120"> </font></b></td>
<td width="41%" bordercolor="#CCCCCC" bgcolor="#FFFFFF"><p
align="center">&Alpha;&nu;&delta;&rho;&iota;&alpha;&nu;ή
&Gamma;&alpha;&lambda;&iota;&alpha;&tau;&sigma;ά&tau;&omicron;&upsilon;
&Alpha;&Mu;:34420</p>
<p align="center"><strong>&Sigma;&Tau;&Epsilon;&Phi;
&Tau;&Mu;&Eta;&Mu;&Alpha;
&Alpha;&Upsilon;&Tau;&Omicron;&Mu;&Alpha;&Tau;&Iota;&Sigma;&Mu;&Omicron;&Upsil
on;</strong></p>
<p align="center"><strong>&Alpha;&Tau;&Epsilon;&Iota;
&Pi;&Epsilon;&Iota;&Rho;&Alpha;&Iota;&Alpha;</strong></p></td>
<td width="40%" bgcolor="#FFFFFF"> <p
align="center">&Pi;&Tau;&Upsilon;&Chi;&Iota;&Alpha;&Kappa;&Eta;
&Epsilon;&Rho;&Gamma;&Alpha;&Sigma;&Iota;&Alpha; </p>
<p align="center">&Delta;&Iota;&Alpha;&Tau;&Alpha;&Xi;&Eta;
&Epsilon;&Lambda;&Epsilon;&Gamma;&Chi;&Omicron;&Upsilon; &Kappa;&Alpha;&Iota;
&Alpha;&Pi;&Omicron;&Mu;&Alpha;&Kappa;&Rho;&Upsilon;&Sigma;&Mu;&Epsilon;&Nu;&E
ta;&Sigma; &Delta;&Iota;&Alpha;&Chi;&Epsilon;&Iota;&Rho;&Iota;&Sigma;&Eta;&Sigma;
&Mu;&Epsilon;&Sigma;&Omega;
&Delta;&Iota;&Alpha;&Delta;&Iota;&Kappa;&Tau;&Upsilon;&Omicron;&Upsilon;
&Alpha;&Upsilon;&Tau;&Omicron;&Mu;&Alpha;&Tau;&Iota;&Sigma;&Mu;&Omega;&Nu;
&Epsilon;&Gamma;&Kappa;&Alpha;&Tau;&Alpha;&Sigma;&Tau;&Alpha;&Sigma;&Eta;&Sig
ma;
&Theta;&Epsilon;&Rho;&Mu;&Omicron;&Kappa;&Eta;&Pi;&Iota;&Omicron;&Upsilon;</p><
/td>
</tr>
</table>

<br>

<table width="675" border="0" cellpadding="0" cellspacing="0">
<tr>
<td width="160" height="122" valign="top">
<p align="center">Stack version: <span
id="txtStackVersion">Unknown</span> <font color="#0000FF"></font><br>
```

```
                                    Build date: <span
id="txtBuildDate">Unknown</span> <br>
                                    <a href="helpfile.htm">HelpFile</a></p>
                                <p align="center"> </p></td>
                                 <td width="641" valign="top"
bordercolor="#CCCCCC">
                                <table width="576" cellpadding="3"
bordercolor="#CCCCCC">
                                <form>
                                 <tr>
                                  <td
width="105"><b>&Epsilon;&nu;&#941;&rho;&gamma;&epsilon;&iota;&epsilon;&sigmaf;
</b></td>
                                  <td width="1"> </td>
                                 </tr>
                                 <tr>
                                  <td
colspan="2">Ε&lambda;&epsilon;&gamma;&chi;&omicron;&sigmaf;
&Epsilon;&xi;ό&delta;&omega;&nu;</td>
                                  <td colspan="3">
                                   <div align="center">
                                    <input type="button" value="RELAY1"
onClick="GetServerFile('0?2=LED5','')">
                                    <input type="button" value="RELAY2"
onClick="GetServerFile('0?3=LED6','')">
                                    <input type="button" value="XR1"
onClick="GetServerFile('0?5=XR1','')">
                                    <input type="button" value="XR2"
onClick="GetServerFile('0?6=XR2','')">
                                    <input type="button" value="XR3"
onClick="GetServerFile('0?7=XR4','')">
                                    <input type="button" value="XR4"
onClick="GetServerFile('0?8=XR5','')">
                                    <input type="button" value="XR5"
onClick="GetServerFile('0?9=XR6','')">
                                    <input type="button" value="XR6"
onClick="GetServerFile('0?0=XR6','')">
                                   </div></td></tr>
                                 <tr>
                                  <td colspan="2"> </td>
                                  <td width="414"><div align="center">
                                   LCD
```

```
                    <input type="text" name="3" id="LCDText" size="32">
                     <input                type="button"                value="Write"
onClick="GetServerFile('1?3='+document.getElementById('LCDText').value,'')">
                     <input                type="button"                value="LIGHT"
onClick="GetServerFile('0?4=LED3','')">
                    </div></td>
                   </tr>
                  </form>
                  <tr>
                   <td                    height="25"                    colspan="2"
bordercolor="#CCCCCC"><b>&Kappa;&alpha;&tau;&#940;&sigma;&tau;&alpha;&sigma;
&eta;</b></td>
                  </tr>
                 </table>                    </td>
                              </tr>
          </table>

                        <span                                          align="center"
id="txtAutoUpdateStatus">Loading...</span>
   </body>
</html>
```

status:

```
                <table width="561" cellpadding="3">
                              <tr>
                                    <td    width="102"    bgcolor="#CCCCCC"><span
class="style3                 style2"><strong>&Alpha;&nu;&tau;&lambda;&iota;&alpha;
:</strong></span></td>
                                    <td     width="61"     bgcolor="#FFFF66"><span
class="style4 style1">%13</span></td>
                                    <td    width="130"    bgcolor="#CCCCCC"><span
class="style3
style2"><strong>&Alpha;&nu;&epsilon;&mu;&iota;&sigma;&tau;&#x1F40;&rho;&alpha;&sigmaf;:
</strong></span></td>
                                    <td width="61" bgcolor="#FFFF66">%27</td>
                                    <td    width="101"    bgcolor="#CCCCCC"><span
class="style3 style2"><strong> EXT_IN1:</strong></span></td>
```

```
                    <td width="54" bgcolor="#FFFF66">%21</td>
                </tr>
                <tr>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Pi;ό&tau;&iota;&sigma;&mu;&alpha;:</strong></span></td>
                    <td bgcolor="#FFFF66"><span class="style4
style1">%14</span></td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Mu;&pi;ό&iota;&lambda;&epsilon;&rho;:</strong></span></td>
                    <td bgcolor="#FFFF66">%28</td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong> EXT_IN2:</strong></span></td>
                    <td bgcolor="#FFFF66">%22</td>
                </tr>
                <tr>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Epsilon;i&sigma;&omicron;&delta;&omicron;&sigmaf;
1:</strong></span></td>
                    <td bgcolor="#FFFF66"><span class="style4
style1">%00</span></td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong> &Epsilon;&xi;&omega;&tau;&epsilon;&rho;&iota;&kappa;ό
&Rho;&epsilon;&lambda;&alpha;i3:</strong></span></td>
                    <td bgcolor="#FFFF66">%29</td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong> EXT_IN3:</strong></span></td>
                    <td bgcolor="#FFFF66">%23</td>
                </tr>
                <tr>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Epsilon;i&sigma;&omicron;&delta;&omicron;&sigmaf;
2:</strong></span></td>
                    <td bgcolor="#FFFF66"><span class="style4
style1">%01</span></td>
                    <td
bgcolor="#CCCCCC"><strong>&Epsilon;&xi;&omega;&tau;&epsilon;&rho;&iota;&kappa;ό
&Rho;&epsilon;&lambda;&alpha;i4:</strong></td>
                    <td bgcolor="#FFFF66">%2A</td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong> EXT_IN4:</strong></span></td>
                    <td bgcolor="#FFFF66">%24</td>
                </tr>
```

```
                        <tr>
                            <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Epsilon;i&sigma;&omicron;&delta;&omicron;&sigmaf;
3:</strong></span></td>
                            <td bgcolor="#FFFF66"><span class="style4
style1">%10</span></td>
                            <td
bgcolor="#CCCCCC"><strong>&Epsilon;&xi;&omega;&tau;&epsilon;&rho;&iota;&kappa;ό
&Rho;&epsilon;&lambda;&alpha;i5:</strong></td>
                            <td bgcolor="#FFFF66">%2B</td>
                            <td bgcolor="#CCCCCC"><span class="style3
style2"><strong> EXTAINV:</strong></span></td>
                            <td bgcolor="#FFFF66">%34</td>
                        </tr>
                        <tr>
                            <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Epsilon;i&sigma;&omicron;&delta;&omicron;&sigmaf;
4:</strong></span></td>
                            <td bgcolor="#FFFF66"><span class="style4
style1">%11</span></td>
                            <td
bgcolor="#CCCCCC"><strong>&Epsilon;&xi;&omega;&tau;&epsilon;&rho;&iota;&kappa;ό
&Rho;&epsilon;&lambda;&alpha;i6:</strong></td>
                            <td bgcolor="#FFFF66">%2C</td>
                            <td bgcolor="#CCCCCC"><span class="style3
style2"><strong> EXTAINI:</strong></span></td>
                            <td bgcolor="#FFFF66">%35</td>
                        </tr>
                        <tr>
                          <td bgcolor="#CCCCCC">&#x0D;&#x0A</td>
                          <td bgcolor="#FFFF66"> </td>
                            <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Theta;&epsilon;&rho;&mu;&omicron;&kappa;&rho;&alpha;&sigma;i&al
pha;:</strong></span></td>
                            <td bgcolor="#FFFF66"><span class="style4
style1">%03</span></td>
                            <td colspan="2" bgcolor="#CCCCCC"><div
align="center">'O&rho;&iota;&omicron;
&Theta;&epsilon;&rho;&mu;&omicron;&kappa;&rho;&alpha;&sigma;i&alpha;&sigmaf;</di
v></td>
                        </tr>
                        <tr>
```

```
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Phi;&lambda;&omicron;&tau;&eacute;&rho; 2:</strong></span></td>
                    <td bgcolor="#FFFF66"><span class="style4
style1">%0D</span></td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Phi;&lambda;&omicron;&tau;&eacute;&rho; 1:</strong></span></td>
                    <td bgcolor="#FFFF66"><span class="style4
style1">%04</span></td>
                    <td bgcolor="#CCCCCC"><span class="style3
style2"><strong>&Pi;&omicron;&tau;&epsilon;&nu;&sigma;&iota;&oacute;&mu;&epsilon;&tau;&r
ho;&omicron;:</strong></span></td>
                    <td bgcolor="#FFFF66"><span class="style4
style1">%02</span></td>
                </tr>


</table>
```

Επίλογος-Συμπεράσματα

Στόχος της πτυχιακής ήταν να καταφέρουμε να ελέγχουμε αυτοματοποιημένα τη θερμοκρασία του θερμοκηπίου και να τη διατηρούμε στα επιθυμητά πλαίσια σύμφωνα με τις ανάγκες της συγκεκριμένης καλλιέργειας καθώς επίσης να ποτίζεται η καλλιέργεια αυτόματα σε συγκεκριμένα χρονικά διαστήματα, τα οποία τα ορίζουμε εμείς. Και τέλος ολόκληρη τη διαδικασία να μπορούμε να την παρακολουθούμε και να την ελέγχουμε από μακριά. Με την υλοποίηση και ολοκλήρωση της εφαρμογής ο στόχος μας επιτεύχθηκε. Επιπλέον την συγκεκριμένη εφαρμογή μπορούμε κάλλιστα να την εφαρμόσουμε και σε οποιαδήποτε άλλη καλλιέργεια, με οποιεσδήποτε απαιτήσεις και αν έχει αυτή, επιτυχώς. Το βασικό πλεονέκτημα της πλακέτας είναι ότι διαθέτει ενσωματωμένο Web server μέσου του οποίου μπορούμε να πραγματοποιούμε απομακρυσμένη επίβλεψη και τηλεέλεγχο της εγκατάστασης από τις εισόδους και εξόδους αντίστοιχα της πλακέτας μέσω οποιουδήποτε κοινού προγράμματος περιήγησης . Λόγω αυτού του πλεονεκτήματος η πλακέτα μπορεί να χρησιμοποιηθεί και σε οποιαδήποτε άλλη εφαρμογή απαιτεί παρακολούθηση από μακριά.

Βιβλιογραφία

www.123rf.com/photo_8274465_drip-irrigation.html

forusv/pic-maxi-web.html

www.olimex.com/dev/pic-icd2.html

1tee-trikal.tri.sch.gr/.../Thermokipio.htm

http://www.cyprus.gov.cy/moa/agriculture.nsf/All/18EA3C203AAACF62C2257667002DF86B/$file/01.%20ARDEYSH.pdfBF

http://el.wikipedia.org/wiki/%CE%98%CE%B5%CF%81%CE%BC%CE%BF%CE%BA%CE%AE%