- **ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΑΤΡΩΝ**
- **ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ**
- **ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΙΑΣ**

- **ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

• ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΕ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ C ΓΙΑ ΔΙΑΧΕΙΡΗΣΗ ΑΠΟΘΕΜΑΤΟΣ ΠΕΤΡΕΛΑΙΟΥ ΘΕΡΜΑΝΣΗΣ

ΣΠΟΥΔΑΣΤΗΣ : ΜΠΑΛΛΙΟΣ ΔΗΜΗΤΡΙΟΣ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ : ΛΟΥΚΑΣ ΧΑΔΕΛΛΗΣ

ΠΑΤΡΑ 2011

ΠΡΟΛΟΓΟΣ

Στην σημερινή εποχή όπου οι ανάγκες του ανθρώπου για ενέργεια αυξάνονται συνεχώς και οι πηγές ενέργειας που χρησιμοποιούμε ως τώρα έχουν αρχίσει να περιορίζονται ως προς το διαθέσιμο υπόλοιπο τους επιβάλετε να αλλάξει η νοοτροπία που επικρατεί μέχρι τώρα σχετικά με την ενέργεια , κυρίως στο θέμα της εξοικονόμησης ενέργειας.

Οι κτιριακές εγκαταστάσεις που υπάρχουν στην Ελλάδα είναι ως επί το πλείστον παλαιές κατασκευές με υπερβολικά μεγάλες καταναλώσεις ενέργειας ανά m^2 .Τα τελευταία χρόνια έχει αρχίσει μια προσπάθεια ενεργειακής αναβάθμισης των ήδη υπαρχόντων κτιρίων έτσι ώστε και οι ανάγκες των ανθρώπων να καλύπτονται στο έπακρο και τα κτίρια να καταναλώνουν ελάχιστη ενέργεια για την λειτουργία τους .

Όσον αφορά την κατανάλωση ηλεκτρικής ενέργειας στα κτίρια η αναβάθμιση τους είναι μια σχετικά εύκολη υπόθεση , αφού οι όποιες μετατροπές μπορούν να γίνουν εύκολα και χωρίς να απαιτούν σημαντικές αλλαγές στις κτιριακές εγκαταστάσεις . Επίσης στον τομέα της ηλεκτρικής ενέργειας η μέτρηση της και ο τρόπος για βέλτιστη απόδοση υπάρχει και έχει εξελιχθεί πάρα πολύ εδώ και αρκετές δεκαετίες.

Ένας άλλος ενεργοβόρος τομέας είναι η ψύξη και η θέρμανση των κτιρίων . Λόγο των χαμηλών προδιαγραφών που ίσχυαν μέχρι πρόσφατα τα περισσότερα κτίρια χρειάζονται πάρα πολύ ενέργεια για την ψύξη και την θέρμανση γιατί δεν διαθέτουν τις σωστές μονώσεις στους τοίχους και κυρίως στα εξωτερικά κουφώματα .Έτσι έχει αρχίσει η διαδικασία μέτρησης των απωλειών των κτιρίων και στην συνέχεια γίνονται οι απαραίτητες επεμβάσεις για την ενεργειακή τους αναβάθμιση.

ΠΕΡΙΛΗΨΗ

Σκοπός τις συγκεκριμένης πτυχιακής είναι η ανάπτυξη μιας εφαρμογής, λογισμικό ,που θα μπορεί να εκτελείτε είτε από έναν προσωπικό υπολογιστή είτε από έναν μικροεπεξεργαστή ο οποίος θα βρίσκετε ενσωματωμένος στο σύστημα θέρμανσης και θα εκτελεί όλες τις απαραίτητες ενέργειες που έχουν να κάνουν με την διαχείριση του πετρελαίου θέρμανσης .

Το σύστημα λαμβάνει πληροφορίες μέσω αισθητήρων πίεσης και αντιλαμβάνεται το περιεχόμενο τις δεξαμενής δηλαδή την ποσότητα καυσίμου που υπάρχει στην δεξαμενή .Η εφαρμογή δέχεται αρχικά από τον χρήστη σαν είσοδο τις φυσικές διαστάσεις τις δεξαμενής, μήκος, πλάτος, για να μπορεί να αντιστοιχήσει την πίεση που δέχεται ο αισθητήρας σε λίτρα καύσιμου. Το σύστημα καταγράφει τις μεταβολές στην ποσότητα κατά την λειτουργία του καυστήρα και υπολογίζει την κατανάλωση σε λίτρα ανά ώρα .

Επίσης μέσω τις καταγραφής μπορούμε να ξέρουμε πόσα λίτρα βάλαμε, αν έχουμε διαρροή ή κλοπή, πότε γεμίσαμε και πόσο, με τι ρυθμό γεμίζει ή αδειάζει, πότε έχουμε πρόβλημα στο καυστήρα ελέγχοντας την κατανάλωση του. Από το αρχείο τις καταγραφής μπορούμε να έχουμε μια συνολική εικόνα τις κατανάλωσης καθ'όλην την διάρκεια του έτους με σκοπό να μπορέσουμε να καταλάβουμε ποιες ενέργειες μπορούμε να κάνουμε ώστε να ρυθμίσουμε την εγκατάσταση μας έτσι ώστε να έχουμε την μέγιστη απόδοση με την ελάχιστη δυνατή κατανάλωση. Σε ένα ολοκληρωμένο σύστημα που θα έχουμε την απόδοση του καυστήρα , των σωμάτων και των υπόλοιπων τμημάτων τις εγκατάστασης εύκολα μπορούμε να υπολογίσουμε πόση ενέργεια και χρήμα ξοδεύουμε για να θερμάνουμε το κτίριο και πόση ενέργεια πάει χαμένη .

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η σωστή διαχείριση της κατανάλωσης ενέργειας στα κτίρια για την θέρμανση τους είναι ένα σημαντικό θέμα. Για να μπορούμε να έχουμε σωστή διαχείριση πρέπει να έχουμε πλήρη έλεγχο στην όλη διαδικασία από την ποσότητα καυσίμου που καταναλώθηκε μέχρι το ποσό τις ενέργειας που έφτασε τελικά στον χώρο .
Η παρακάτω εργασία αφορά το κομμάτι τις διαχείρισης του πετρελαίου που χρησιμοποιείτε για την θέρμανση κτιρίων. Αρχικά γίνετε μια αναφορά στις δεξαμενές που χρησιμοποιούνται στις περισσότερες εφαρμογές τα υλικά που κατασκευάζονται και το σχήμα που επιλέγετε ανάλογα με την χωρητικότητά τους. Στην συνέχεια έχουμε τους διάφορους αισθητήρες που μετατρέπουν σε ψηφιακό σήμα το φυσικό μέγεθος τις στάθμης του πετρελαίου. Έπειτα έχουμε τον μικροεπεξεργαστή και τέλος το πιο σημαντικό τμήμα της εργασίας που είναι το software που τρέχει στον μικροεπεξεργαστή και στην ουσία παίρνει τα δεδομένα από τον αισθητήρα τα επεξεργάζεται και εμφανίζει τον τρέχοντα όγκο της δεξαμενής και αποθηκεύει στοιχειά σε εξωτερική μνήμη .Ένα σημαντικό πλεονέκτημα είναι ότι το software μπορεί να χρησιμοποιηθεί σε οποιοδήποτε μικροεπεξεργαστή που υποστηρίζει .NET Micro Framework και αν κριθεί απαραίτητο μπορούν να γίνουν πολλές προσθήκες στο πρόγραμμα πολύ εύκολα.

1. ΔΕΞΑΜΕΝΕΣ ΠΕΤΡΕΛΑΙΟΥ

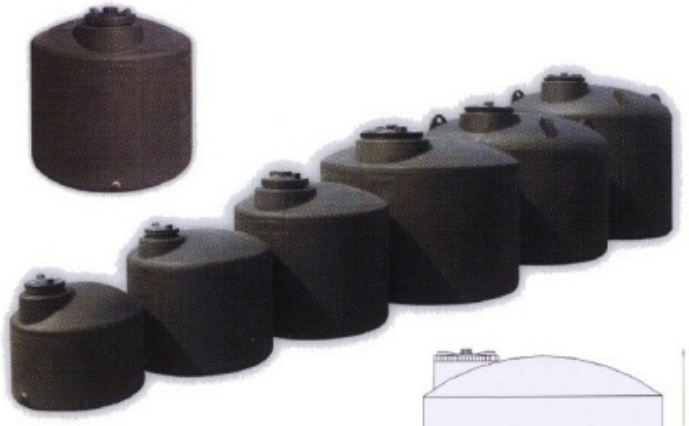
Στις εγκαταστάσεις θέρμανσης ένα βασικό στοιχείο είναι η δεξαμενή αποθήκευσης του πετρελαίου .Ανάλογα με το κτίριο και τις απαιτήσεις σε αυτονομία αλλάζει και ο όγκος που μπορούν να χωρέσουν, σε οικιακές εγκαταστάσεις οι συνήθεις χωρητικότητες είναι από 500L έως 2000L .Εκτός από την χωρητικότητα τους δυο άλλα χαρακτηριστικά γνωρίσματα των δεξαμενών είναι το σχήμα τους και το υλικό κατασκευής τους. Ανάλογα με το σχήμα τους έχουμε τις παρακάτω διαφορές.

1.1 Δεξαμενές κυλινδρικές κατακόρυφες

Είναι πλαστικές δεξαμενές με χωρητικότητες που ξεκινούν από 500L και μπορούν να φτάσουν μέχρι και τα 52000L ανάλογα με την εφαρμογή που θα χρησιμοποιηθούν. Λόγο του σχήματος τους αυτές οι δεξαμενές δεν διογκώνονται όταν έχουν υγρό και αυτό τις καθιστά ιδανικές όταν πρόκειται να χρησιμοποιηθεί αισθητήρας υπερήχων στην κορυφή της δεξαμενής ή αισθητήρας μέτρησης πίεσης κατακόρυφης στήλης υγρού για τον προσδιορισμό του όγκου του υγρού που υπάρχει στο εσωτερικό της.

● ΔΥΝΑΤΟΤΗΤΑ ΤΟΠΟΘΕΤΗΣΗΣ ΣΕ ΣΤΕΝΟΥΣ ΧΩΡΟΥΣ.

ΛΙΤΡΑ LITRES	ΔΙΑΣΤΑΣΕΙΣ - DIMENSIONS (m)		
	ΔΙΑΜΕΤΡΟΣ DIAMETER	ΥΨΟΣ HEIGHT	ΔΙΑΜ. ΚΑΠ. LID'S DIAM.
650	0,75	1,71	0,22
850	0,82	1,78	0,22
1.000	0,80	2,01	0,22
1.050	0,90	1,85	0,22
1.500	1,03	2,00	0,22
1.800	1,13	1,99	0,22
2.100	1,15	2,10	0,35
3.200	1,61	1,70	0,35



- ΑΠΟ 3.200 LT
  ΜΕΓΑΛΟ ΚΑΠΑΚΙ (Φ 550)
  ΓΙΑ ΕΙΣΟΔΟ ΚΑΙ ΚΑΘΑΡΙΣΜΟ.

- ΜΙΚΡΟ ΥΨΟΣ ΓΙΑ ΟΠΤΙΚΟ ΕΛΕΓΧΟ.

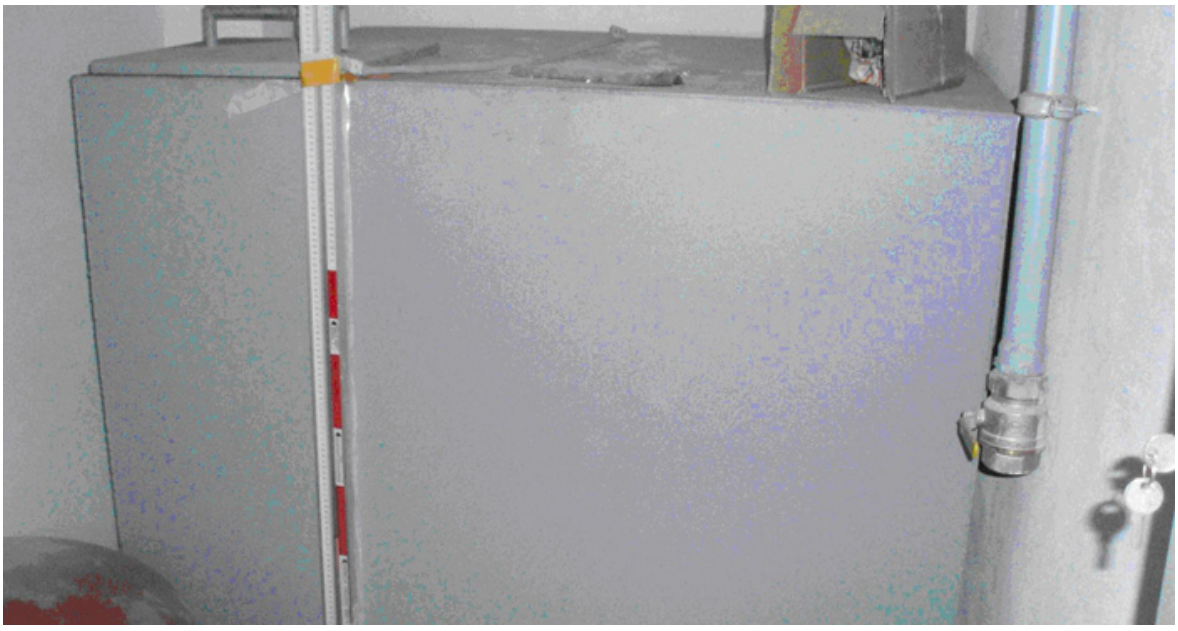
ΛΙΤΡΑ LITRES	ΔΙΑΣΤΑΣΕΙΣ - DIMENSIONS (m)		
	ΔΙΑΜΕΤΡΟΣ DIAMETER	ΥΨΟΣ HEIGHT	ΔΙΑΜ. ΚΑΠ. LID'S DIAM.
500	0,95	0,85	0,22
600	1,02	0,88	0,22
800	1,12	0,98	0,22
1.000	1,18	1,11	0,35
1.250	1,25	1,25	0,35
1.500	1,31	1,32	0,35
2.000	1,50	1,36	0,35
3.200	1,87	1,41	0,55
4.000	2,18	1,39	0,55
5.000	2,18	1,63	0,55
6.200	2,20	1,94	0,55
7.000	2,39	1,87	0,55
8.000	2,39	2,08	0,55

1.2 Δεξαμενή οριζόντια ελλειψοειδής

Εδώ έχουμε έναν άλλο τύπο δεξαμενής ο οποίος έχει το μειονέκτημα ότι για να γίνει μέτρηση του όγκου λόγο του ότι δεν είναι γραμμική η μεταβολή του ύψους του υγρού σε σχέση με τον όγκο που υπάρχει στο εσωτερικό του ,πρέπει να γνωρίζουμε από πριν τη σχέση ύψους όγκου ,σχέση την οποία οφείλει να γνωστοποιεί ο κατασκευαστής της δεξαμενής στον εκάστοτε αγοραστή.

1.3 Δεξαμενές μεταλλικές

Είναι δεξαμενές τετράγωνες ή ορθογώνιες οι οποίες έχουν το πλεονέκτημα ότι συναρμολογούνται στο σημείο όπου θα τοποθετηθούν .Όπως και οι στρόγγυλες πλαστικές δεξαμενές δεν διογκώνονται όταν είναι γεμάτες υγρό και αυτό βοηθάει στην περίπτωση που θα χρησιμοποιηθεί αισθητήρας υπερήχων ή μέτρησης πίεσης γιατί μπορεί να γίνει εύκολα η αναγωγή του ύψους του υγρού σε λίτρα χωρίς να χρειάζεται κάποιος ιδιαίτερος υπολογισμός για τον ακριβή προσδιορισμό του.



2. ΜΕΤΡΗΣΗ ΣΤΑΘΜΗΣ

Για τον προσδιορισμό του όγκου του υγρού των δεξαμενών χρησιμοποιούνται διάφοροι τύποι αισθητήρων με διαφορετική αρχή λειτουργίας . Ενδεικτικά αναφέρονται διάφοροι τρόποι μέτρησης.

Χωρητικότητα (RF)

Μαγνητοπεριοριστική

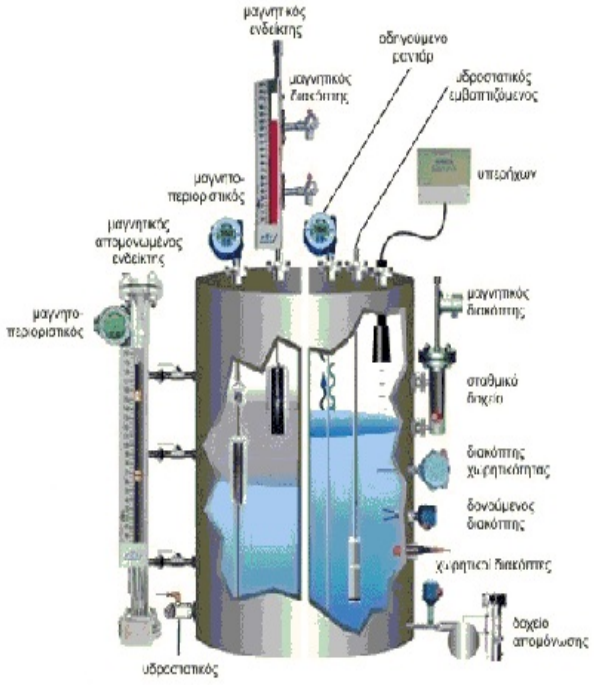
Μαγνητική

Υδροστατική

Ραντάρ

Υπερήχων

Λέηζερ



Οι πιο συνήθεις για μικρές δεξαμενές και χωρίς μεγάλες πιέσεις είναι μετρώντας την πίεση του υγρού με αισθητήρα πίεσης (μέτρηση υδροστατικής πίεσης στήλης υγρού), μέτρηση ύψους στάθμης υγρού με πομπό και δέκτη υπερήχων.

2.1 Αισθητήρες Στάθμης Υπερήχων

Οι αισθητήρες υπερήχων παρέχουν μια οικονομικά αποδοτική μέθοδο ανίχνευσης με ιδιότητες που δεν υπάρχουν σε άλλες τεχνολογίες. Με τη χρήση μιας ευρείας ποικιλίας μετατροπέων υπερήχων και διάφορα φάσματα συχνοτήτων, ένας αισθητήρας υπερήχων μπορεί να σχεδιαστεί για να λύσει πολλά προβλήματα εφαρμογών που είναι απαγορευτικά στο κόστος ή απλά δεν μπορούν να λυθούν από άλλους αισθητήρες. Οι αισθητήρες υπερήχων εκπέμπουν συνεχώς ηχητικούς παλμούς υψηλής συχνότητας προς την επιφάνεια του στόχου και ανακλώνται πίσω στον αισθητήρα. Τα ηλεκτρονικά του αισθητήρα μετρούν το χρόνο λήψης του σήματος και τον μετατρέπουν σε μονάδα μήκους. Από εδώ υπολογίζεται ο όγκος του υγρού που υπάρχει στην προς μέτρηση δεξαμενή.

Δεδομένου ότι η ταχύτητα του ήχου επηρεάζεται από τη θερμοκρασία αέρα, οι αισθητήρες υπερήχων περιλαμβάνουν έναν ενσωματωμένο αισθητήρα θερμοκρασίας. Οι μετρήσεις στάθμης/απόστασης αντισταθμίζονται αυτόματα σε όλη την κλίμακα λειτουργίας του αισθητήρα.

2.2 Αισθητήρες πίεσης

Οι αισθητήρες πίεσης είναι μια φθηνή και αξιόπιστη λύση για πάρα πολλές εφαρμογές είτε αυτό έχει να κάνει με μέτρηση , έλεγχο κτλ. Ένας τέτοιος αισθητήρας τοποθετημένος στο πυθμένα της δεξαμενής αντιλαμβάνεται την πίεση που ασκεί το υγρό στην επιφάνεια του και δεδομένου ότι γνωρίζουμε πόση είναι αυτή η επιφάνεια και ότι 1 λίτρο πετρελαίου ζυγίζει 0,83 K gr υπολογίζουμε το ύψος της στήλης του υγρού που υπάρχει από πάνω του . Στη συνέχεια μπορούμε εύκολα να ανάγουμε το

ύψος του υγρού της δεξαμενής στον συνολικό όγκο του υγρού αφού γνωρίζουμε όλες τις διαστάσεις της δεξαμενής.

Αισθητήρας πίεσης

3. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C#

Η C# είναι αναμφίβολα η γλώσσα προγραμματισμού που επιλέγετε για το .NET περιβάλλον.
Είναι μια καινούρια γλώσσα ελεύθερη από προβλήματα προηγούμενων εκδόσεων με πάρα πολλές νέες δυνατότητες. Είναι αντικειμενοστραφείς γλώσσα προγραμματισμού (Object Oriented Programming language) και έχει στον πυρήνα της πολλές ομοιότητες με την Java, C++ και VB. Η C# συνδυάζει την ισχύ και την αποδοτικότητα τις C++ , τον απλό και σαφή OO σχεδιασμό τις Java και την απλότητα τις VB.

3.1 Οι βασικές ιδέες

- Component-orientation
- Καθετί είναι ένα αντικείμενο
- Εύρωστο και συντηρήσιμο λογισμικό
- Υποστήριξη διαλειτουργικότητας

Η C# είναι η πρώτη “Component-Oriented” γλώσσα στην οικογένεια C/C++

- Τι είναι ένα component?
- Μια ανεξάρτητη οντότητα που μπορεί να επαναχρησιμοποιηθεί
- Περιλαμβάνει πολλαπλές κλάσεις
- Συχνά ανεξάρτητο από γλώσσες

Component-Orientation

Έννοιες component

- Properties, methods, events

- Reflection API
- Design-time και run-time attribute

Καθιστά δυνατό το “one-stop programming”

- Δεν υπάρχουν header files, IDL, κλπ.
- Ενσωμάτωση σε ASP σελίδες

Παραδοσιακά

C++, Java™: Δεν υποστηρίζεται διαλειτουργικότητα μεταξύ βασικών τύπων και αντικειμένων

Smalltalk, Lisp: Οι βασικοί τύποι είναι αντικείμενα αλλά με σημαντικό κόστος στην απόδοση

Η C# παρέχει ενοποίηση χωρίς κόστος στην απόδοση
Απλότητα σε όλο το σύστημα
Βελτιωμένη επεκτασιμότητα και επαναχρησιμοποίηση
Νέοι βασικοί τύποι: Decimal, SQL…

Εύρωστο και συντηρήσιμο λογισμικό

Garbage collection
Καμία διαρροή μνήμης, ξεκρέμαστοι δείκτες
Εξαιρέσεις
Βελτιωμένος χειρισμός σφαλμάτων
Type-safety
Versioning
Αποφυγή συχνών σφαλμάτων
Π.χ. if (x = y) ...

Υποστήριξη διαλειτουργικότητας

C++ Κληρονομιά
Namespaces, pointers, unsigned types, etc.
Διαλειτουργικότητα
Η C# μπορεί να χρησιμοποιηθεί σε συνδυασμό με XML, SOAP, COM, DLLs, και οποιαδήποτε γλώσσα του.NET Framework
Αυξημένη παραγωγικότητα
Ευκολία εκμάθησης

4. Το .NET Micro Framework

Το .NET Micro Framework είναι bootable runtime module που φέρνει τα πλεονεκτήματα του. NET προγραμματισμού για συσκευές με πολύ περιορισμένους πόρους. Τα οφέλη της ανάπτυξης με το. NET Micro Framework περιλαμβάνουν την

γλώσσα προγραμματισμού C #, ένα διαχειριζόμενο περιβάλλον εκτέλεσης, ένα σημαντικό υποσύνολο βιβλιοθηκών του .NET , και Visual Studio ™ για την ανάπτυξη και τον εντοπισμό σφαλμάτων.
Παρόλο που έχει μέγεθος μόνο 300KB το .NET Micro Framework παρέχει ένα πλήρες περιβάλλον διαχείρισης εκτέλεσης με αυτόματη διαχείριση μνήμης, ένα σημαντικό υποσύνολο του. NET Base Class Library, GUI κατηγορίες με βάση το Windows Presentation Foundation, ένα διαχειριζόμενο μοντέλο οδήγησης συσκευής , διατηρούμενη αποθήκευση, και υποστήριξη για πολλά ενσωματωμένα πρότυπα για τη διεπαφή για σειριακή επικοινωνία και δικτύωση.
Ένα σημαντικό επίτευγμα του .NET Micro Framework όπως και οι άλλες .NET πλατφόρμες
είναι ότι κατάφεραν να είναι ανεξάρτητες από το υλικό στο οποίο θα εγκατασταθούν,αυτό κάνει πολύ εύκολη την μεταφορά του λογισμικού σε άλλες hardware πλατφόρμες.
Όχι πολύ καιρό πριν, τα ενσωματωμένα συστήματα χρησιμοποιούσαν τους ίδιους επεξεργαστές και τις ίδιες τις γλώσσες προγραμματισμού όπως οι γενικής χρήσης μικροϋπολογιστές . Οι πρώτοι μικροεπεξεργαστές στην πραγματικότητα, έχουν σχεδιαστεί για χρήση σε ενσωματωμένες εφαρμογές. Λόγο περιορισμένης επεξεργαστικής ισχύς και αποθηκευτικής ικανότητας των μικροεπεξεργαστών , το λογισμικό της εποχής ήταν γραμμένο σε συμβολική γλώσσα, μια χρονοβόρα και επιρρεπής σε λάθη διαδικασία που απαιτεί υψηλού επιπέδου δεξιότητες προγραμματισμού.

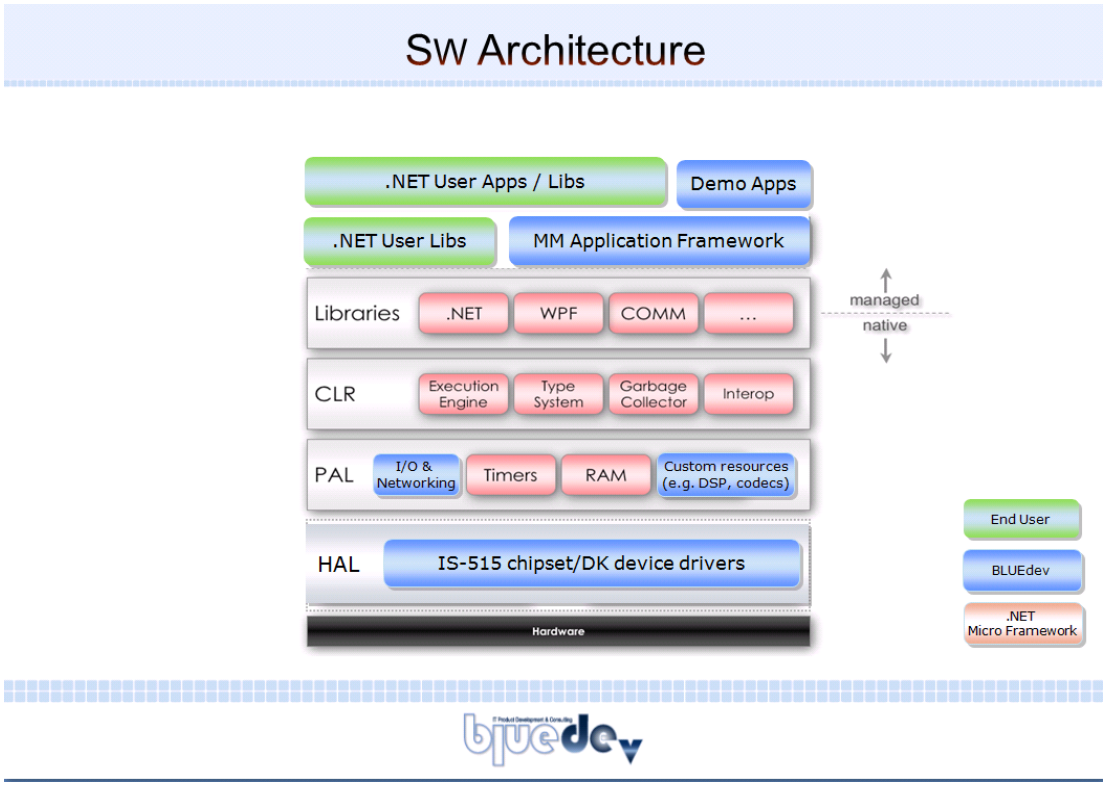
Καθώς οι προσωπικοί υπολογιστές έγιναν εκθετικά πιο ισχυροί, εργαλεία προγραμματισμού ακολούθησαν το παράδειγμα, προωθώντας την παραγωγικότητα προγραμματισμού υψηλότερα. Ανάμεσα στις πιο σημαντικές καινοτομίες είναι:

- Όλο και πιο εκφραστικές υψηλού επιπέδου γλώσσες προγραμματισμού
- Ολοκληρωμένο περιβάλλον ανάπτυξης (IDEs) με γραφικά περιβάλλοντα εργασίας χρήστη
- Ευρέως διαθέσιμες υψηλού επιπέδου βιβλιοθήκες
- Διαχείριση περιβάλλοντος εκτέλεσης με αυτόματη συλλογή των απορριμμάτων (garbage collection)

Το. NET Micro Framework , είναι η μικρότερη. NET πλατφόρμα μέχρι σήμερα, φέρνει τα οφέλη του. NET και άλλα σύγχρονα παραδείγματα τη σύγχρονη γλώσσα C# , ένα διαχειριζόμενο περιβάλλον εκτέλεσης, και το παγκόσμιας κλάσης Visual Studio ® αναπτυξιακό εργαλείο. Τώρα κάθε. NET προγραμματιστής μπορεί επίσης να είναι προγραμματιστής embedded συστημάτων.

Τα κυριότερα σημεία του .NET Micro Framework είναι :

- The smallest .NET footprint yet (about 300 KB of RAM)
- A version of the .NET class library tailored to embedded applications, including GUI classes modeled on the Windows Presentation Foundation (WPF)
- A bootable CLR that can run directly on hardware without an operating system
- Support for common hardware and interconnects (nonvolatile memory, GPIO, I2C, RS232, SPI)
- Managed device driver model for devices connected via the supported interconnects
- Seamless persistent storage through extended weak references
- Full Visual Studio integration, including live debugging of code running on a tethered device
- An extensible emulator that lets you test applications in a window on your PC



Εικόνα 1: Αρχιτεκτονική λογισμικού βασισμένη στο .Net Micro Framework

Ενώ το κόστος παραγωγής ανά μονάδα θα είναι πάντα ύψιστης σημασίας, για κάποιες ενσωματωμένες εφαρμογές, οι προγραμματιστές συχνά διαπιστώνουν ότι το χαμηλότερο κόστος ανάπτυξης και ο γρηγορότερος χρόνος εξόδου στην αγορά, σε συνδυασμό με την αξιοπιστία, επεκτασιμότητα και τα πλεονεκτήματα που προσφέρει. NET, μπορεί να αντιστάθμιση το υψηλότερο κόστος BOM, που απαιτεί ενα hardware για .NET Micro Framework .

Το. NET Micro Framework δεν είναι μόνο για να φέρουμε ενσωματωμένες εφαρμογές πιο γρήγορα στην αγορά. Από τα έξυπνα σπίτια για οικιακές συσκευές, από δίκτυα αισθητήρων οι ενσωματωμένες εφαρμογές γίνονται όλο και πιο εξελιγμένες από ποτέ. Νέα προσιτά, χαμηλής ισχύος ασύρματα πρωτόκολλα δικτύωσης που επιτρέπουν σε μια νέα κατηγορία μικρών συσκευών απρόσκοπτα την ανταλλαγή δεδομένων με άλλες ασύρματες συσκευές εξοπλισμένες όποτε βρίσκονται σε σειρά, εάν οι άλλες συσκευές είναι διακομιστές, σταθμούς εργασίας επιφάνεια εργασίας, έξυπνα τηλέφωνα ή PDAs, ή άλλες μικρές συσκευές.

Το .NET Micro Framework είναι ιδανική λύση για χομπίστες που αναζητούν έναν γρήγορο, αξιόπιστο τρόπο για την ανάπτυξη προσαρμοσμένων ελεγκτων υλικού για μια ποικιλία εφαρμογών. Ένα πλήρως εξοπλισμένο κιτ ανάπτυξης για ένα. NET Micro Framework κόστιζει μόλις μερικές εκατοντάδες ευρω (συμπεριλαμβανομένων έγχρωμη οθόνη LCD) και κάνει ενα απίστευτα ευέλικτο ελεγκτή για όλα τα είδη προσωπικών έργων. Τα περισσότερα κιτ ανάπτυξης έχουν σειριακή και γενικής χρήσης εισόδου / εξόδου (GPIO) διεπαφές που διευκολύνουν την διασύνδεση της συσκευής με κινητήρες, αισθητήρες, και άλλο υλικό. Και με το Visual Studio, το. NET Micro Framework είναι επίσης ένας εξαιρετικός τρόπος για να εισαγάγει τους φοιτητές στη διασκέδαση του προγραμματισμού με τη συμμετοχή τους στον πραγματικό κόσμο, τα έργα, όπως η ρομποτική, εργαστήριακα όργανα, και κοινωνικές εφαρμογές.

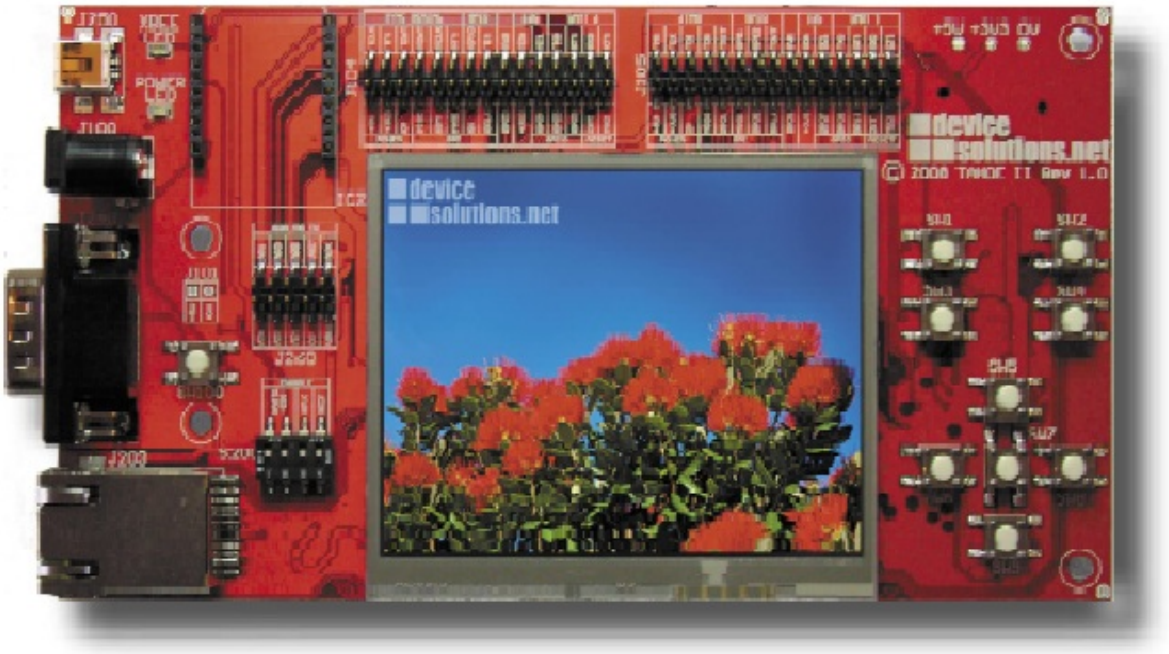
4.1 Απαιτήσεις υλικού

Επί του παρόντος το .NET Micro Framework τρέχει σε διάφορες εκδοσεις των ARM7 και ARM9 μικροεπεξεργαστών που διατίθενται από μεγάλους προμηθευτές υλικού. Οι προτεινόμενες ελάχιστες απαιτήσεις μνήμης (χωρίς να υπολογίζουμε τις ανάγκες εφαρμογής) είναι 300 KB RAM και 512K MB μνήμης flash. Η συσκευή πρέπει επίσης να έχει θύρα USB, ή δίκτυο να υποστηρίζει τη λήψη και τον εντοπισμό σφαλμάτων των εφαρμογών.

5. Το Development Board Tahoe II

Στη συγκεκριμένη πτυχιακή χρησιμοποιήθηκε το αναπτυξιακό board Tahoe II το οποίο συλλέγει τα δεδομένα από τον αισθητήρα υπολογίζει τον όγκο που υπάρχει στην δεξαμενή και αποθηκεύει δεδομένα στην SD κάρτα μνήμης.

Μια περιγραφή του board ακολουθεί παρακάτω.



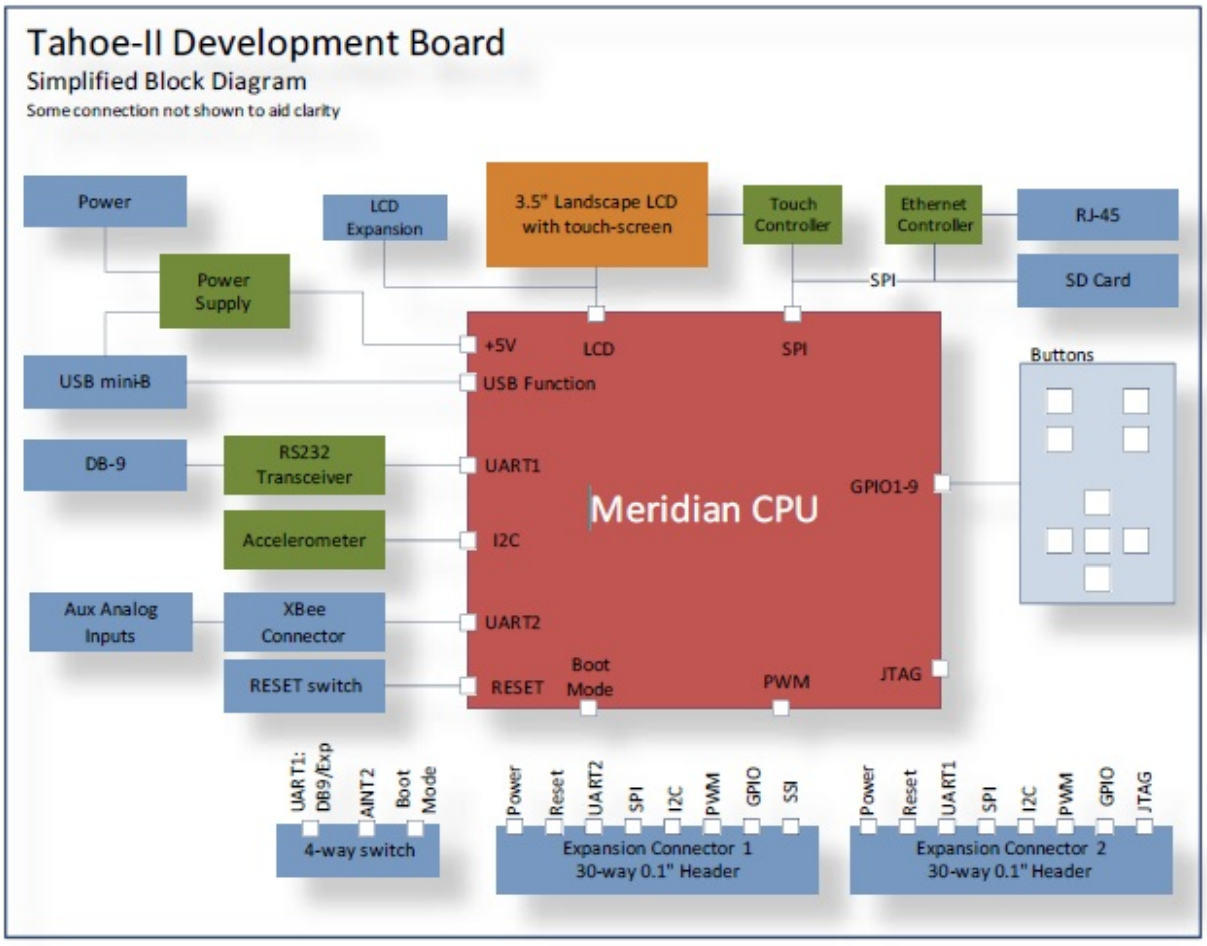
Εικόνα 2: Development Board Tahoe II

Χαρακτηριστικά:
• 3.5” LCD with touch-screen
• Ethernet
• USB Function for application download and debug
• Accelerometer
• Serial ports
• SD card support
• XBee module connector for wireless communications

Debug με το Visual-Studio απευθείας στην συσκευή.
Χρησιμοποιώντας την USB σύνδεση κατεβάζουμε κώδικα και τον τρέχουμε στο board.

Πολυάριθμες εφαρμογές
• Automotive displays
• Building automation and access control
• Home energy management
• Windows® SideShow™-enabled displays and remote controls
• Desktop/handheld instrumentation
• Industrial monitoring and control
• Medical devices
• Retail signage and information kiosks

Ο πυρήνας του είναι:
• Meridian CPU (i.MXS @ 100MHz, 4MBytes Flash, 8MBytes SDRAM)



6. Η UML

Τι είναι μοντέλο ;

- Αποτελεί μια **αφηρημένη περιγραφή** ενός φυσικού συστήματος.
- Αποτελεί ένα σχέδιο για την **κατασκευή** ενός συστήματος.
- Βοηθάει στην **κατανόηση** ενός μεγάλου μεγέθους συστήματος.
- Βοηθάει στην **επικοινωνία** των μελών της ομάδας που αναπτύσσει το σύστημα.

Τι είναι η UML ;

Η UML είναι μια οπτική αντικειμενοστρεφής γλώσσα μοντελοποίησης που χρησιμοποιείται για:

- Απεικόνιση (visualization)
- Προδιαγραφή (specification)
- Τεκμηρίωση (documentation)
- Κατασκευή (construction)

των δομικών συστατικών ενός συστήματος (λογισμικού ή όχι).

Χαρακτηριστικά της **UML**

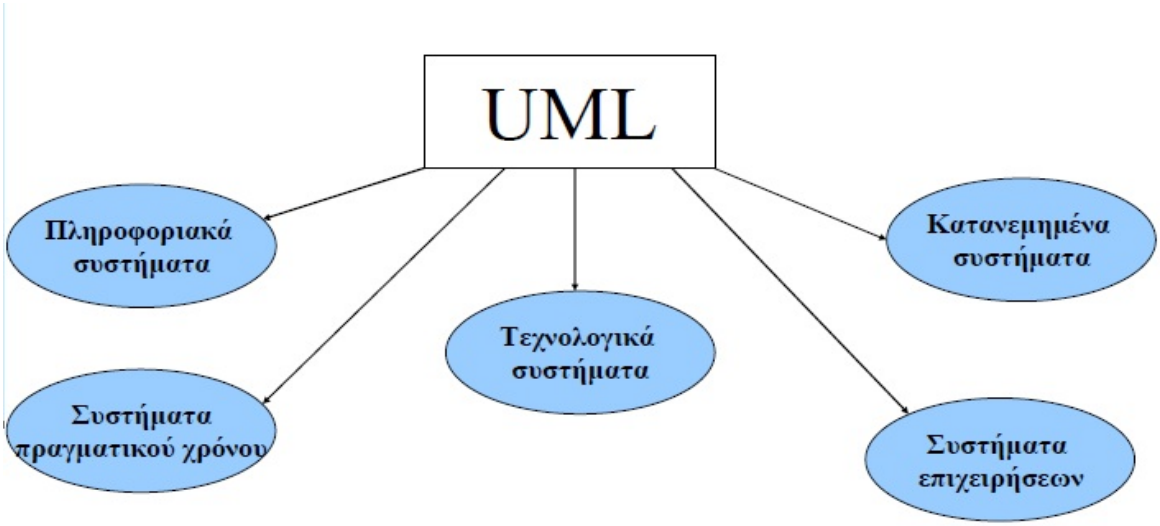
• Είναι ιδιαίτερα εκφραστική.
• Σημασιολογικά, είναι εκτενής.
Υποστηρίζει τη σημασιολογία τύπων και μοντέλων για όλα τα μοντέλα ενός συστήματος.
• Σε επίπεδο βασικών αρχών, είναι “μικρή” και απλή.
Διακρίνουμε πέντε βασικούς άξονες.
• Είναι επεκτάσιμη.
Υπάρχει η δυνατότητα εμπλουτισμού του μετα-μοντέλου με κλάσεις, ιδιότητες και σημασιολογία.
• Είναι επακριβώς ορισμένη με βάση τα δομικά συστατικά ενός αντικειμενοστρεφούς συστήματος.
• Προήλθε από την ενοποίηση των συμβολισμών που χρησιμοποιούσαν οι μεθοδολογίες Booch, OMT, OOSE κ.λπ.
• Ενσωματώνει τις ιδέες “καλής πρακτικής” από τη βιομηχανία λογισμικού.
• Είναι σήμερα βιομηχανικό πρότυπο και αναπτύχθηκε στο πλαίσιο του οργανισμού OMG.
• Υλοποιεί την ανάγκη της βιομηχανίας λογισμικού για μια ενιαία γλώσσα μοντελοποίησης.
• Αντιμετώπιση σημερινών και βραχυπρόθεσμων προβλημάτων στην ανάπτυξη λογισμικού:

- Κλίμακα
- Γλώσσες: Java, C++, Smalltalk, Ada, Visual Basic
- Πολυεπεξεργασία και παραλληλία
- Πρότυπα Λογισμικού (Patterns)
- Ψηφίδες λογισμικού (Componentware)
- Μοντελοποίηση επιχειρησιακής πρακτικής

Είδη συστημάτων που μοντελοποιούνται με τη UML

• Συστήματα με έμφαση στο λογισμικό (software-intensive systems)
• Επιχειρησιακά συστήματα (business systems)
• Συστήματα που δεν περιέχουν λογισμικό (non-software systems)

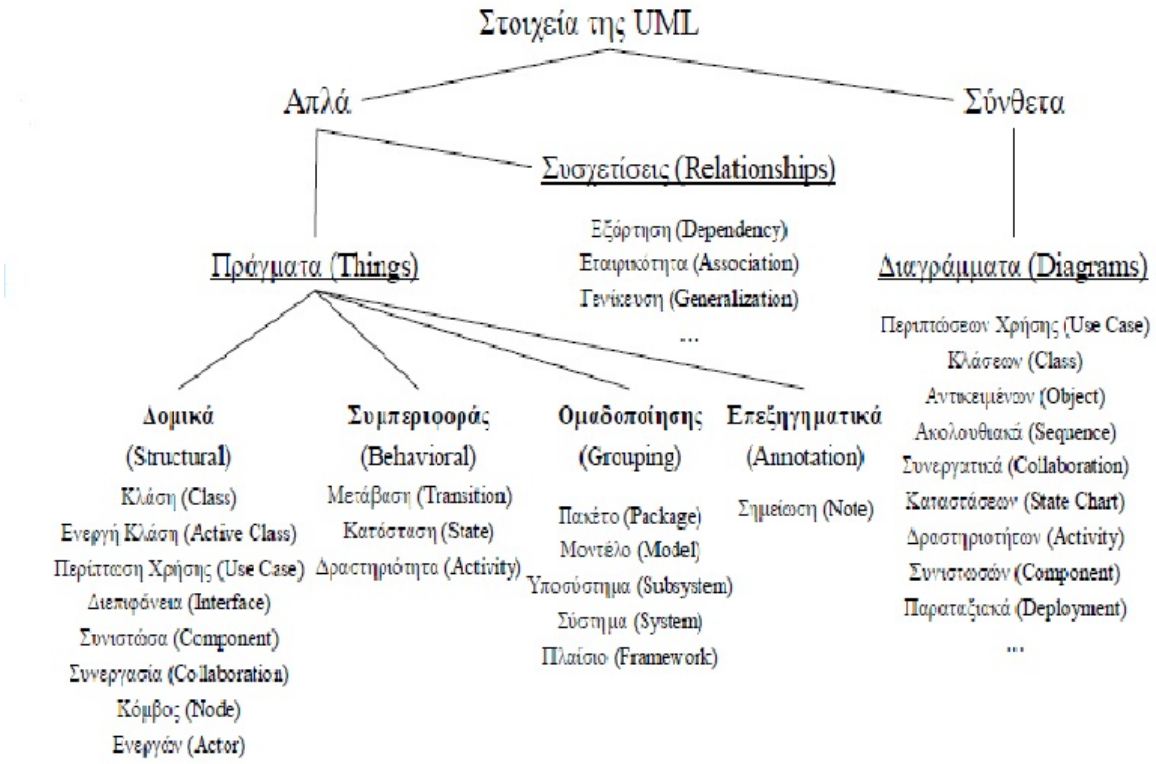
Χρήση της UML



Οι πέντε βασικοί άξονες της **UML**

• Στοιχεία του μοντέλου (model elements)
• Συσχετίσεις (relationships)
• Μηχανισμοί (mechanisms)
• Διαγράμματα (diagrams)
• Αρχιτεκτονικές όψεις (architectural views)

Μια κατηγοριοποίηση των στοιχείων της **UML**



Δομικά στοιχεία της **UML (I)**

• Κλάση (class)
Ενα σύνολο αντικειμένων με κοινή δομή και συμπεριφορά.
• Ενεργή κλάση (active class)
Μια κλάση που περιγράφει μια διεργασία ή ένα νήμα εκτέλεσης και αλληλεπιδρά με άλλες.
• Περίπτωση χρήσης (use case)
Μια λειτουργία που επιτελεί ένα σύστημα και είναι διαθέσιμη στο χρήστη. Είναι συμπεριφορά του συστήματος που συνεπάγεται τη συνεργασία ενός συνόλου αντικειμένων.
• Διεπιφάνεια (interface)
Ένα σύνολο από λειτουργίες που ορίζουν την εξωτερική συμπεριφορά ενός αντικειμένου.
• Συνιστώσα (component)
Ένα φυσικό και επαναχρησιμοποιήσιμο τμήμα ενός συστήματος, με λογική και φυσική υπόσταση που συνήθως υλοποιεί κάποιες διεπιφάνειες (interfaces).
• Συνεργασία (collaboration)
Η περιγραφή μιας διάδρασης μεταξύ ενός συνόλου αντικειμένων.

• Κόμβος (node)
Ένας υπολογιστικός πόρος που έχει κάποια μνήμη και υπολογιστική ικανότητα, οπότε εκεί αποθηκεύεται ή/και εκτελείται το λογισμικό.
• Ενεργών (actor)
Εξωτερική του συστήματος οντότητα που χρησιμοποιεί τη λειτουργικότητα και τις διεπιφάνειές του.

Στοιχεία συμπεριφοράς στη **UML**

• Κατάσταση (state)
Μια συνθήκη ή περίπτωση στο χρόνο ζωής ενός αντικειμένου, όπου ικανοποιεί κάποιους περιορισμούς, εκτελεί κάποια δραστηριότητα ή αναμένει κάποιο γεγονός.
• Μετάβαση (transition)
Μια σχέση μεταξύ δύο καταστάσεων ενός αντικειμένου που υποδηλώνει αλλαγή στην κατάσταση του αντικειμένου με την εμφάνιση ενός γεγονότος.
• Δραστηριότητα (activity)
Μια εκτέλεση λειτουργίας κατά κατά τη διάρκεια ζωής ενός αντικείμενου.

Στοιχεία ομαδοποίησης στη **UML**

• Πακέτο (package)
Ένα δομικό στοιχείο γενικής χρήσης για την οργάνωση άλλων δομικών στοιχείων, διαγραμμάτων ή και άλλων πακέτων της UML σε ομάδες.
• Υποσύστημα (subsystem)
Μια μονάδα στην ιεραρχική αποσύνθεση ενός μεγάλου συστήματος. Επικοινωνεί με το περιβάλλον του μέσω διαπροσωπειών.
• Μοντέλο (model)
Μια όψη του συστήματος.

Επεξηγηματικά στοιχεία της **UML**
• Σημείωση (note)
Ένα δομικό στοιχείο κειμενικού σχολιασμού για την περιγραφή ή επεξήγηση ενός άλλου δομικού στοιχείου ή μιας ομάδας δομικών στοιχείων. Αποτελεί σχόλιο ή επεξήγηση ή κείμενο αναφοράς.

Συσχετίσεις στη **UML (I)**
• Συσχέτιση (Association)
Μια δομική σχέση που περιγράφει ένα σύνολο συνδέσεων μεταξύ αντικειμένων.
• Εξάρτηση (Dependency)
Μια σχέση μεταξύ δυο δομικών στοιχείων, όπου μια αλλαγή στο πρώτο επιδρά στο δεύτερο.
• Γενίκευση (Generalization)
Μια σχέση μεταξύ ενός δομικού στοιχείου (πατέρας) και ενός δεύτερου (παιδί) που εξειδικεύει το πρώτο.
Συσχετίσεις στη **UML (II)**

• Συσσωμάτωση (Aggregation)
Μια σχέση μεταξύ δυο δομικών στοιχείων, όπου το πρώτο μπορεί να περιέχει το δεύτερο.
• Σύνθεση (Composition)
Μια σχέση μεταξύ δυο δομικών στοιχείων, όπου το πρώτο εντάσσεται αναπόσπαστα στο δεύτερο και δεν μπορεί να ανήκει σε κανένα άλλο του ιδίου τύπου.
• Πραγματοποίηση (Realization)
Μια σχέση μεταξύ δυο δομικών στοιχείων, όπου το πρώτο προδιαγράφει μια συμπεριφορά και το δεύτερο την υλοποιεί.

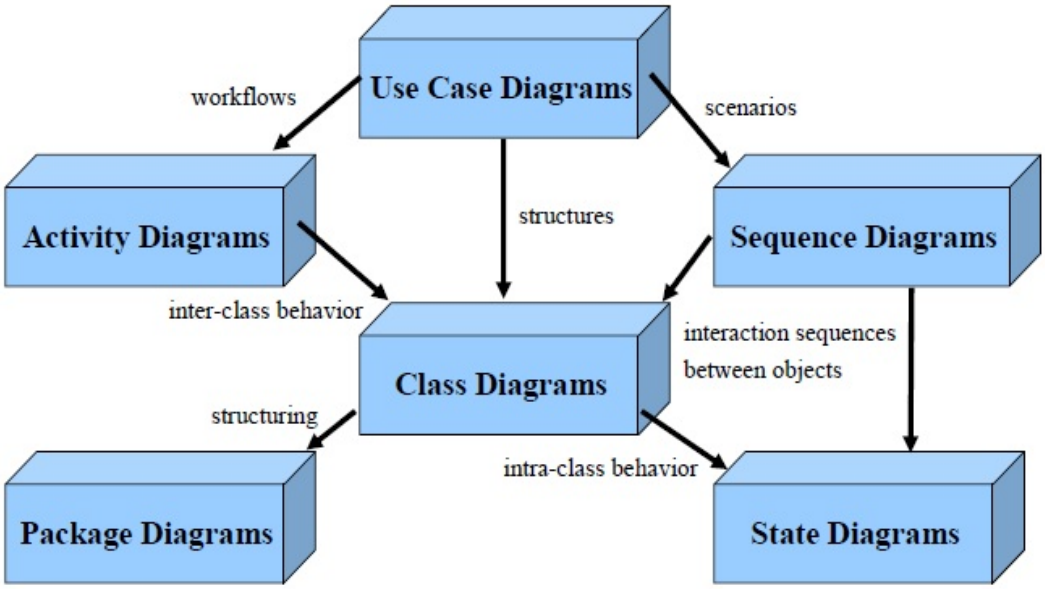
Τύποι διαγραμμάτων στη **UML**

• Δομικά Διαγράμματα (**Structural Diagrams**): Περιγράφουν την εσωτερική λογική δομή
ενός συστήματος, δηλαδή τα συστατικά του και τις σχέσεις μεταξύ τους.
Διάγραμμα Κλάσεων (**Class Diagram**)
Διάγραμμα Αντικειμένων (**Object Diagram**)
Διάγραμμα Συνιστωσών (**Component Diagram**)
Παραταξιακό διάγραμμα (**Deployment Diagram**)

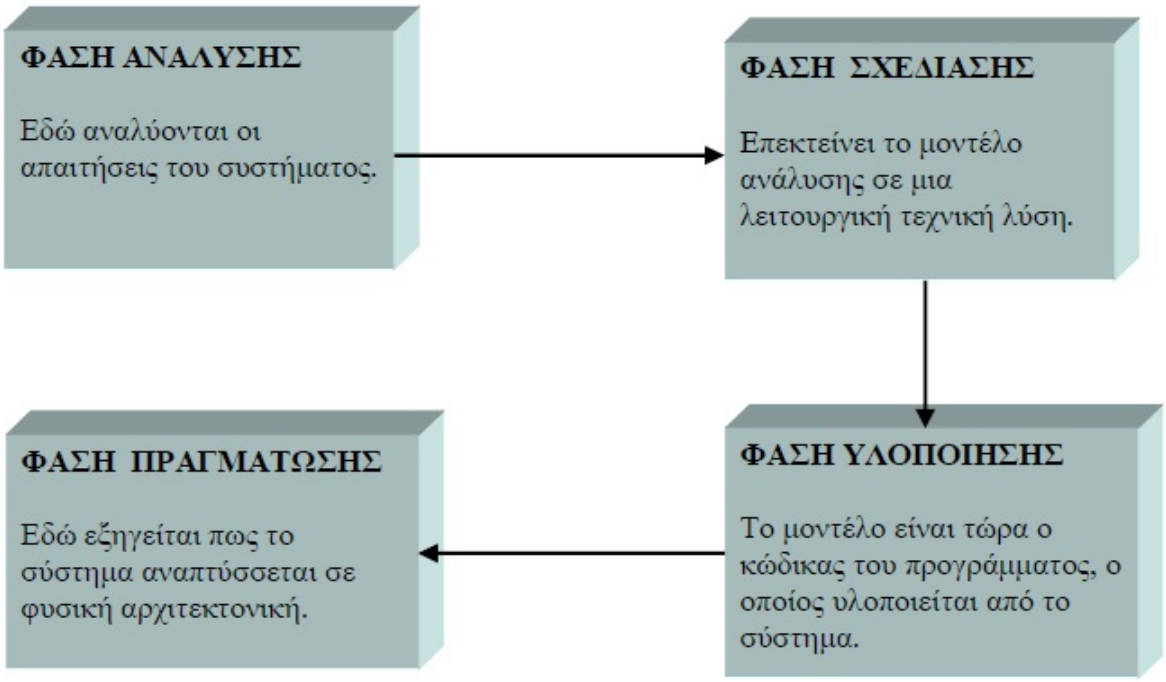
• Διαγράμματα Συμπεριφοράς (**Behavior Diagrams**): Περιγράφουν τη δυναμική συμπεριφορά ενός συστήματος, δηλαδή την απόκρισή του σε γεγονότα του περιβάλλοντός του.
Διάγραμμα Περιπτώσεων Χρήσης (**Use Case Diagram**)
Διάγραμμα Αλληλουχίας (**Sequence Diagram**)
Διάγραμμα δραστηριοτήτων (**Activity Diagram**)
Διάγραμμα Συνεργασίας (**Collaboration Diagram**)
Διάγραμμα Καταστάσεων (**Statechart Diagram**)

• Διαγράμματα Διαχείρισης Μοντέλου (**Model Management Diagrams**):
Περιγράφουν τη φυσική δομή ενός συστήματος, δηλαδή τις μονάδες λογισμικού που το
αποτελούν, σε όρους περιβάλλοντος υλοποίησης.
Διάγραμμα Πακέτων (**Package Diagram**)
Διάγραμμα Υποσυστημάτων (**Subsystem Diagram**)
Διάγραμμα Μοντέλων (**Model Diagram**)

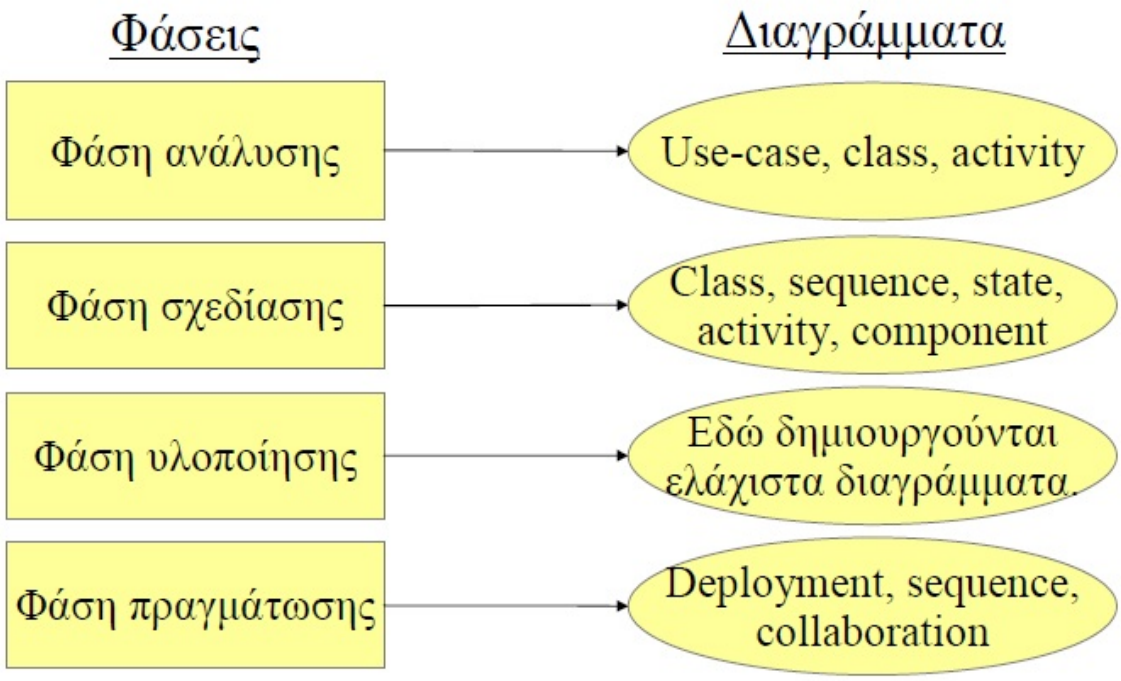
Σχέσεις διαγραμμάτων στη **UML**



6.1 Μοντελοποιώντας με την UML



Αντιστοιχία διαγραμμάτων στις διάφορες φάσεις ανάπτυξης συστήματος



Λειτουργίες των εργαλείων μοντελοποίησης **(*Case Tools*) (I)**

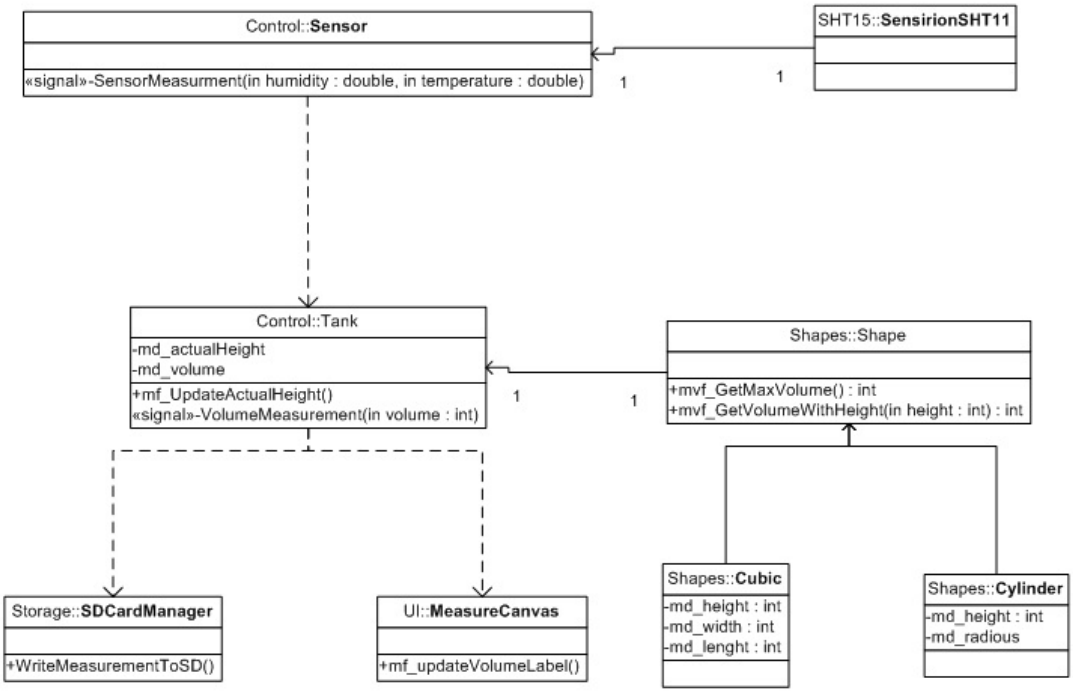
- Σχεδιασμός διαγραμμάτων που εξασφαλίζουν την σωστή χρήστη των στοιχείων του μοντέλου.
- Αποθήκευση δεδομένων. Αν το όνομα μιας κλάσης αλλάζει σ' ένα διάγραμμα, η αλλαγή αυτή διαδίδεται και στα υπόλοιπα.
- Υποστήριξη διάδοσης ενός στοιχείου από ένα διάγραμμα σε ένα άλλο κι επέκταση της περιγραφής του στοιχείου.
- Υποστήριξη πολλών χρηστών. ίνει την δυνατότητα να δουλεύουν πολλοί χρήστες ταυτόχρονα.
- Δημιουργία κώδικα

Λειτουργίες των εργαλείων μοντελοποίησης **(*Case Tools*) (II)**

- Αντίστροφη παραγωγή, δηλαδή δυνατότητα παραγωγής μοντέλων από τον κώδικα (reverse engineering).
- Συμβατότητα με άλλα εργαλεία όπως editors, μεταγλωττιστές κι άλλα επιχειρησιακά εργαλεία.
- Κάλυψη όλων των επιπέδων ανάπτυξης του συστήματος, από το επίπεδο περιγραφής του συστήματος στο επίπεδο κώδικα.
- Επικοινωνία ανάμεσα στα μοντέλα. Ένα διάγραμμα σε κάποιο μοντέλο θα πρέπει να έχει τη δυνατότητα να εισάγεται από το ένα μοντέλο στο άλλο.

6.2 Το UML διάγραμμα του συστήματος

Αφού έχουμε αποφασίσει τι θέλουμε να κάνει το πρόγραμμά μας το σχεδιάζουμε με χρήση της UML.



Με βάση το παραπάνω διάγραμμα έγινε και η υλοποίηση του κώδικα.

7. ΤΟ ΥΛΙΚΟ ΚΟΜΜΑΤΙ

Από την πλευρά του υλικού μέρους του συστήματος για την υλοποίηση του χρειαζόμαστε τα παρακάτω.

Α) Την προς μέτρηση δεξαμενή:

Καλό είναι ο αισθητήρας να τοποθετείτε δίπλα στην δεξαμενή στο ύψος του πυθμένα της με κάποιο σωλήνα και όχι απευθείας στον πυθμένα για να είναι εύκολος ο έλεγχος και η αν χρειαστεί η αντικατάστασή του.

Β) Έναν αισθητήρα πίεσης

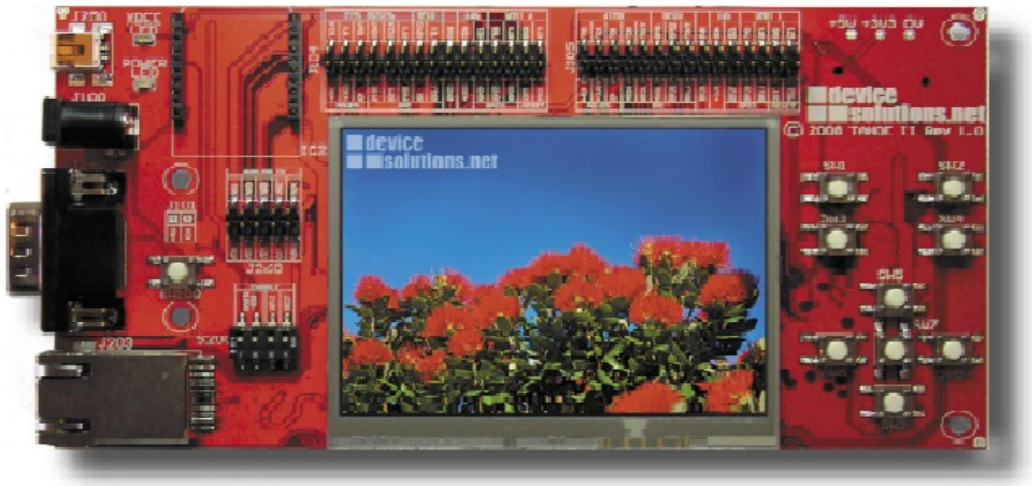
Πλέον τα ολοκληρωμένα κυκλώματα υπάρχουν σχεδόν σε όλους του τύπους και σε όλα τα είδη αισθητήρων . Έτσι έχουμε καταφέρει να έχουμε πολύ μεγάλη ακρίβεια στις μετρήσεις
και πολλές επιλογές στην επικοινωνία με τους μικροεπεξεργαστές όπως SPI , UART ,I2C.

Γ) Τον μικροεπεξεργαστή

Οποιοσδήποτε μικροεπεξεργαστής που μπορεί να υποστηρίξει .NET Micro Framework μπορεί να χρησιμοποιηθεί χωρίς κανένα πρόβλημα .

Ενδεικτικά διάφορα board FEZ που υποστηρίζουν .NET Micro Framework :

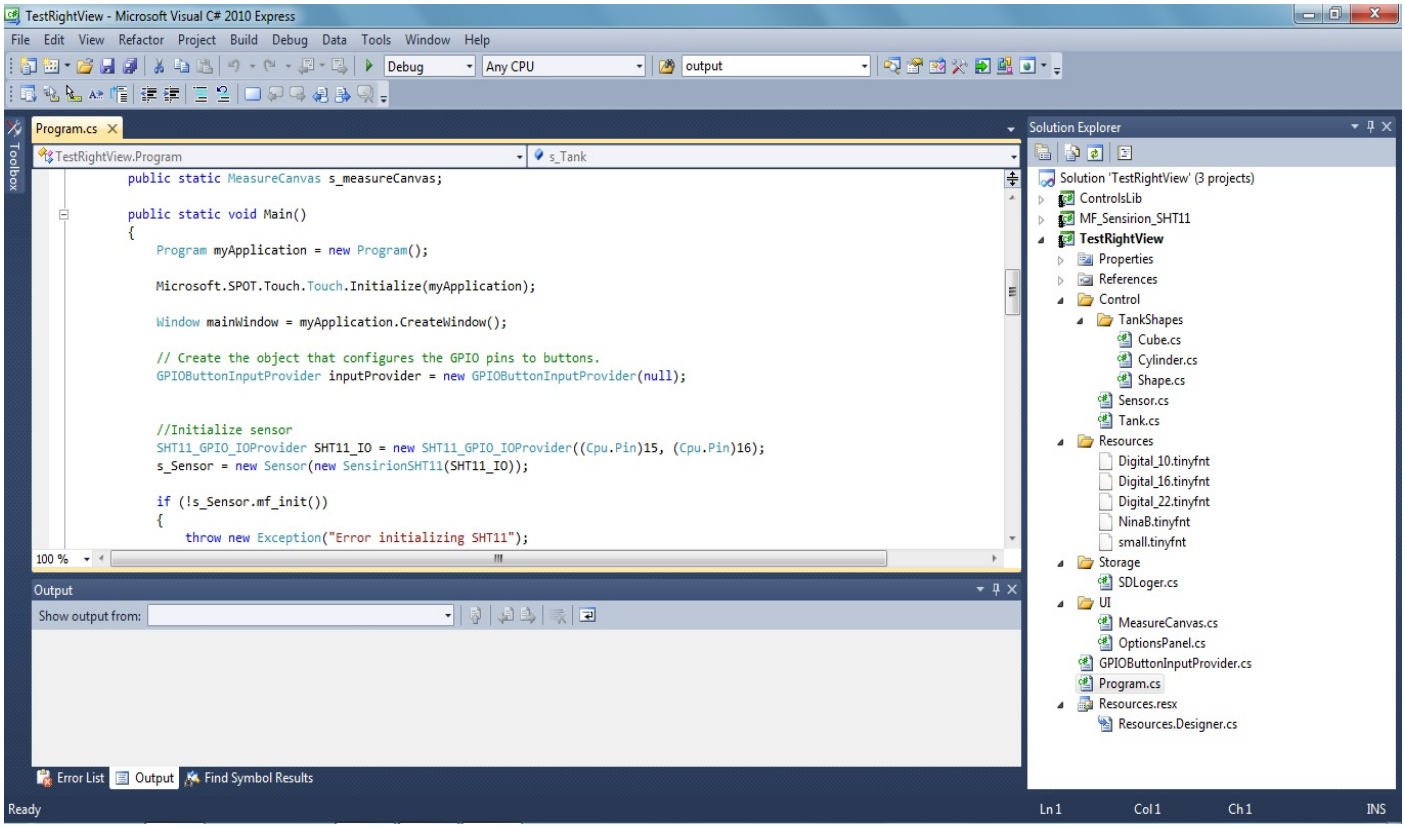
Και αυτό που χρησιμοποιήθηκε στην συγκεκριμένη περίπτωση Tahoe II



Ένα σημαντικό πλεονέκτημα αυτών των συσκευών λόγο του ότι έχουν δημιουργηθεί για το .NET Micro Framework είναι ότι υπάρχει πάρα πολύ μεγάλη υποστήριξη από την πλευρά του λογισμικού για ενημερώσεις και λογισμικό για πάρα πολλές περιφερειακές συσκευές όπως LCD οθόνες , ασύρματη επικοινωνία , πληθώρα αισθητήρων ,συσκευές αποθήκευσης, επικοινωνία με το internet κτλ.

8. ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΕΡΓΑΣΙΑΣ

Ένα πολύ σημαντικό πλεονέκτημα του προγραμματισμού στο .NET Micro Framework είναι το εξαιρετικό Visual-Studio. Είναι από τα πιο καλά IDE που υπάρχουν και προσφέρει ότι χρειάζεται ο προγραμματιστής για να κάνει σωστά και σε πολύ μικρο χρόνο την ανάπτυξη οποιασδήποτε εφαρμογής.



Στο κέντρο έχουμε το περιβάλλον στο οποίο γράφουμε τον κώδικα . Στην δεξιά πλευρά είναι τα διάφορα τμήματα του προγράμματος ,αυτό βοηθάει στο να “σπάσουμε” όλο το πρόγραμμα σε μικρά κομμάτια για να είναι πιο κατανοητό, εύχρηστο και αν υπάρχει κάποιο σφάλμα δεν χρειάζεται να ψάξουμε όλο το πρόγραμμα άπλα επιλέγουμε το συγκεκριμένο τμήμα που μας ενδιαφέρει .
Τέλος στο κάτω μέρος μπορούμε να επιλέξουμε να βλέπουμε την λίστα με τις προειδοποιήσεις και τα σφάλματα που έχει αναγνωρίσει το Visual-Studio καθώς και όποια εντολή εκτελείτε ανά πάσα στιγμή.

9. To Software

Το παρακάτω τμήμα του κώδικα δημιουργεί την πρώτη οθόνη όπου ο χρήστης επιλέγει τον τύπο της δεξαμενής κυλινδρική ή τετράγωνη ,ορθογώνια. Και αφού κάνει την επιλογή του
εισάγει τις διαστάσεις τις σε mm. Αν δεν γίνει εισαγωγή των δεδομένων και να πατήσουμε το OK το σύστημα δεν προχωράει παρακάτω.


```
using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Controls;
using Microsoft.SPOT.Presentation.Media;
using Bluedev.NMF.Controls;

namespace TestRightView
{
    public enum gt_Resolution { R_320_240, R_240_320 };
    enum Shapeselection {NONE , CUBE , CYLINDER};
    public delegate void FinishEventHandler(Object sender, OptionsEventArgs e);

    public class OptionsPanel :Canvas
    {
        private StackPanel md_heightPanel;
        private gc_GUITextBox md_heightTextBox;
```

```
        private StackPanel md_widthPanel;
        private gc_GUITextBox md_widthTextBox;

        private StackPanel md_lenghtPanel;
        private gc_GUITextBox md_lenghtTextBox;

        private StackPanel md_radiousPanel;
        private gc_GUITextBox md_radiusTextBox;

        public event FinishEventHandler Finished;


        private Shapeselection md_shapeselection = Shapeselection.NONE;

        public OptionsPanel()
            :base()
        {

            Font small = Resources.GetFont(Resources.FontResources.Digital_10);


            //TextBox width
            md_widthTextBox = new gc_GUITextBox();
            md_widthPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_widthPanel, md_widthTextBox, "Width ", 1,
10,Visibility.Collapsed);

             //TextBox lenght
            md_lenghtTextBox = new gc_GUITextBox();
            md_lenghtPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_lenghtPanel, md_lenghtTextBox, "Lenght", 29, 10,
Visibility.Collapsed);

            //TextBox radious
            md_radiusTextBox = new gc_GUITextBox();
            md_radiousPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_radiousPanel, md_radiusTextBox, "Radious", 29, 10,
Visibility.Collapsed);

            //TextBox heigh
            md_heightTextBox = new gc_GUITextBox();
            md_heightPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_heightPanel, md_heightTextBox, "Height ", 59, 10,
Visibility.Collapsed);

            //Toggle Tank shape
            gc_GUIToggleManager manager = new gc_GUIToggleManager(new
gc_ToggleManagerStackPanel(Orientation.Vertical), "Shape");
```

```
 //Button Cube
        gc_GUIToggle comboCube = new gc_GUIToggle(new
gc_ViewButtonRadio(15,15),"CUBE");
        comboCube.Click += new EventHandler(combo_Click);


        //Button Cylinder
        gc_GUIToggle comboCylinder = new gc_GUIToggle(new
gc_ViewButtonRadio(15, 15), "CYLINDER");
        comboCylinder.Click += new EventHandler(comboCylinder_Click);

        manager.mf_AddToggle(comboCube);
        manager.mf_AddToggle(comboCylinder);


        Children.Add(manager.mp_GUIView);


        Canvas.SetTop(manager.mp_GUIView, 40);
        Canvas.SetLeft(manager.mp_GUIView, 210);

        //Label Cube
        gc_GUILabel labelCube = new gc_GUILabel("CUBE", small);
        labelCube.mp_Background = null;
        labelCube.mp_Border = null;
        labelCube.mp_ForeColor = Colors.Green;
        labelCube.Width = 70;
        Children.Add(labelCube);
        Canvas.SetTop(labelCube, 40);
        Canvas.SetLeft(labelCube, 230);

        //Label Cylinder
        gc_GUILabel labelCylinder = new gc_GUILabel("CYLINDER", small);
        labelCylinder.mp_Background = null;
        labelCylinder.mp_Border = null;
        labelCylinder.mp_ForeColor = Colors.Green;
        labelCylinder.Width = 70;
        Children.Add(labelCylinder);
        Canvas.SetTop(labelCylinder, 55);
        Canvas.SetLeft(labelCylinder, 230);


        //Keyboard

        gc_GUIKeyBoard hexKeyBoard = new gc_GUIKeyBoard(new
gc_ViewKeyBoardWrapPanel(), md_heightTextBox);
        hexKeyBoard.mf_Register((gi_Editable)md_widthTextBox);
        hexKeyBoard.mf_Register((gi_Editable)md_lenghtTextBox);
        hexKeyBoard.mf_Register((gi_Editable)md_radiusTextBox);
```

```
            ((gc_ViewKeyBoardWrapPanel)hexKeyBoard.mp_GUIView).Width = 120;

            addKey(new Object[] { new gc_KeyInfoChar('1') }, "1", hexKeyBoard,small);
            addKey(new Object[] { new gc_KeyInfoChar('2') }, "2", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('3') }, "3", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('4') }, "4", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('5') }, "5", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('6') }, "6", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('7') }, "7", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('8') }, "8", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('9') }, "9", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('.') }, ".", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoChar('0') }, "0", hexKeyBoard, small);
            addKey(new Object[] { new gc_KeyInfoCommand(KeyCommand.Clear) }, "C",
hexKeyBoard, small);

            Children.Add(hexKeyBoard.mp_GUIView);

            Canvas.SetTop(hexKeyBoard.mp_GUIView, 106);
            Canvas.SetLeft(hexKeyBoard.mp_GUIView, 100);




            //ok Button

            gc_GUIButton applyButton = new gc_GUIButton(new
gc_ViewButtonRender("Ok", Resources.GetFont(Resources.FontResources.small),
45, 25));
            applyButton.Click += OnClick;
            Children.Add(applyButton.mp_GUIView);
            Canvas.SetTop(applyButton.mp_GUIView, 190);
            Canvas.SetLeft(applyButton.mp_GUIView, 250);

        }

        void combo_Click(object sender, EventArgs e)
        {

            md_heightPanel.Visibility = Visibility.Visible;
            md_widthPanel.Visibility = Visibility.Visible;
            md_lenghtPanel.Visibility = Visibility.Visible;
            md_radiousPanel.Visibility = Visibility.Collapsed;


             md_heightTextBox.TextContent = "";
             md_widthTextBox.TextContent = "";
             md_lenghtTextBox.TextContent = "";
             md_radiusTextBox.TextContent = "";
```

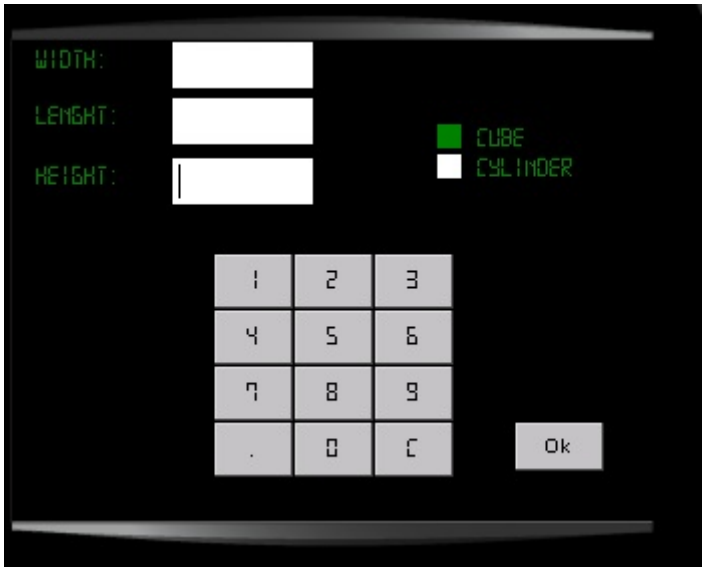
```
            md_shapeselection = Shapeselection.CUBE;

        }

        void comboCylinder_Click(object sender, EventArgs e)
        {

            md_heightPanel.Visibility = Visibility.Visible;
            md_widthPanel.Visibility = Visibility.Collapsed;
            md_lenghtPanel.Visibility = Visibility.Collapsed;
            md_radiousPanel.Visibility = Visibility.Visible;

            md_heightTextBox.TextContent = "";
            md_widthTextBox.TextContent = "";
            md_lenghtTextBox.TextContent = "";
            md_radiusTextBox.TextContent = "";

            md_shapeselection = Shapeselection.CYLINDER;
        }



        private void AddTextBox(StackPanel panel,gc_GUITextBox textBox, String
caption, int top, int left,Visibility visibility)
        {

            Font big = Resources.GetFont(Resources.FontResources.Digital_10);
            Font small = Resources.GetFont(Resources.FontResources.Digital_10);

            //Label
            gc_GUILabel label = new gc_GUILabel(caption +":", big);
            label.mp_Background = null;
            label.mp_Border = null;
            label.mp_ForeColor = Colors.Green;
            label.Width = 70;
            panel.Children.Add(label);

            textBox.Font = big;
            textBox.ForeColor = Colors.Green;
            textBox.Border = new Pen(Colors.White);
            textBox.TextContent = "";
            textBox.CaretIndex = 0;
            textBox.VerticalAlignment = VerticalAlignment.Center;
            textBox.HorizontalAlignment = HorizontalAlignment.Center;
            textBox.Width = 70;
            textBox.Height = 23;
            textBox.OvertypeMode = false;
            panel.Children.Add(textBox);
```

```
            //Add to stackpanel
            Children.Add(panel);
            Canvas.SetTop(panel, top);
            Canvas.SetLeft(panel, left);

            panel.Visibility = visibility;



        }


        private void addKey(object[] objs, String name, gc_GUIKeyBoard keyboard,Font
font)
        {

            gc_GUIKey key = null;

            key = new gc_GUIKey(new gc_ViewButtonRender(name, font, 40, 28),
                        new gc_ModelMultiKey(objs));
            keyboard.mf_AddKey(key);
        }


        private void mf_FireFinished(OptionsEventArgs e)
        {
            if (Finished != null)
                Finished(this, e);
        }

        private void OnClick(Object sender, EventArgs e)
        {
            //TODO check if the textboes are empty before firing the event

            switch (md_shapeselection)
            {
                case Shapeselection.NONE:
                    break;
                case Shapeselection.CUBE:
                    if ((md_heightTextBox.TextContent == "") ||
                        (md_widthTextBox.TextContent == "") ||
                        (md_lenghtTextBox.TextContent == ""))
                        return;
                    mf_FireFinished(new OptionsEventArgs(new
CubeInfo(Convert.ToInt32(md_heightTextBox.TextContent),
                                                        Convert.ToInt32(md_widthTextBox.TextC
ontent),
```

```
   Convert.ToInt32(md_lenghtTextBox.TextContent))));
                    break;
                case Shapeselection.CYLINDER:
                    if ((md_heightTextBox.TextContent == "") ||
                        (md_radiusTextBox.TextContent == ""))
                        return;
                    mf_FireFinished(new OptionsEventArgs(new
CylinderInfo(Convert.ToInt32(md_heightTextBox.TextContent),
                                                         Convert.ToInt32(md_radiusTextBox.Te
xtContent))));
                    break;
                default:
                    break;
            }


        }
    }


    public class OptionsEventArgs : EventArgs
    {
        public ShapeInfo md_info;

        public OptionsEventArgs(ShapeInfo info)
            : base()
        {
            md_info = info;
        }

    }

    public class ShapeInfo { }

    public class CubeInfo : ShapeInfo
    {
        public int md_height;
        public int md_width;
        public int md_lenght;

        public CubeInfo(int height, int width, int lenght)
            : base()
        {
            md_height = height;
            md_width = width;
            md_lenght = lenght;

        }
     }
```

```
    public class CylinderInfo : ShapeInfo
    {
        public int md_height;
        public int md_radius;

        public CylinderInfo(int height, int radius)
            : base()
        {
            md_height = height;
            md_radius = radius;
        }

    }
}
```

Για την περίπτωση που επιλέξουμε τετράγωνη εμφανίζονται 3 μικρά παράθυρα όπου εισάγουμε τις διαστάσεις μήκος ,πλάτος και ύψος σε mm και ο παρακάτω κώδικας υπολογίζει τον όγκο που υπάρχει από τα δεδομένα του αισθητήρα .




```
using System;
using Microsoft.SPOT;

namespace TestRightView.Control
{
    //dimiourgia clasis gia tetragono i or8ogonio sxima
    public class Cube : Shape
    {
        // auto to sxima exei ipsos , mikos kai platos
        private int md_height;
```

```
 private int md_width;
        private int md_lenght;


        public Cube(int height, int width, int lenght)
        {
            md_height = height;
            md_width = width;
            md_lenght = lenght;
        }
        // epistrefei tin megisti xoritikotita tis deksamenis
        public override int mvf_GetMaxVolume()
        {


            return (md_height/10 * md_width/10 * md_lenght/10)/1000;
        }

        //epestrepse ton ogko me ti metrisi tou sensor
        public override int mvf_GetVolumeWithHeight(int height)
        {
            return (height * md_width/10 * md_lenght/10)/1000 ;
        }


    }
}
```

Αν επιλέξουμε κυλινδρική εμφανίζονται 2 παράθυρα όπου εισάγουμε το ύψος και την ακτίνα σε mm και ο παρακάτω κώδικας υπολογίζει τον όγκο :

```
using System;
using Microsoft.SPOT;

namespace TestRightView.Control
{
   //dimiourgia tis clasis gia kylindrikes deksamenes

   public class Cylinder : Shape
   {
      //o kulindros exei aktina kai ipsos
      private int md_height;
      private int md_radius;


      public Cylinder(int height, int radius)
      {
         md_height = height/10;
         md_radius = radius/10;
      }
      //epestrefei ton megisto ogko poy xoraei h deksameni me ya stoixeia pou
dosame
      public override int mvf_GetMaxVolume()
      {
         return (int)(3.14F * md_radius * md_radius * md_height)/1000;
      }
      //epistrefei ton ogko me basi tin metrisi tou sensor
      public override int mvf_GetVolumeWithHeight(int height)
      {


         return (int)(3.14F * md_radius * md_radius   * height )/1000;
      }

   }
}
```

Τα δυο παραπάνω τμήματα υπολογίζουν τον όγκο της δεξαμενής άρα πρέπει να φτιάξουμε μια κλάση που περιγράφει την δεξαμενή. Η δεξαμενή έχει σχήμα και ανάλογα με το σχήμα της έχει ακτίνα και ύψος ή μήκος ,πλάτος και ύψος.

```
using System;
using System.Threading;
using Microsoft.SPOT;

namespace TestRightView.Control
{
   //dimiourgia clasis Tank
```

```
 public class Tank
  {
     public delegate void HaveMeasurementEventHander(Object sender,
VolumeEventArgs e);
     public event HaveMeasurementEventHander HaveMeasurement;


     private Shape md_shape;
     private int md_actualHeight;


     public Tank()
     {
        md_shape = null;
        md_actualHeight = -1;

         Program.s_Sensor.HaveMeasurement +=new
Sensor.HaveMeasurementEventHander(s_Sensor_HaveMeasurement);
     }


     public Tank(Shape shape)
     {
        md_shape = shape;
     }



     private void mf_HaveMeasurement(VolumeEventArgs e)
     {
        if (HaveMeasurement != null)
           HaveMeasurement(this, e);
     }

      private void  s_Sensor_HaveMeasurement(object sender,
MeasurementEventArgs e)
      {
         Control.MeasurementEventArgs measurement =
(Control.MeasurementEventArgs)e;
         //perna tin timi tou sensor san trexon ipsos
          md_actualHeight = (int)measurement.md_temperature;

         VolumeEventArgs ev = new VolumeEventArgs(e.md_temperature,
e.md_humidity, mf_GetActualVolume());
      mf_HaveMeasurement(ev);


      }
```

```
        public void mf_SetShape(Shape shape)
        {
            md_shape = shape;
        }

        //an exei epilegei sxima epestrepse max ogko
        public int mf_GetMaxVolume()
        {
            if (md_shape == null)
                return 0;
            else
                return md_shape.mvf_GetMaxVolume();
        }

        public int mf_GetActualVolume()
        {   // yparxoun data apo ton sensor gia to ipsos ?
            if (md_actualHeight == -1) return md_actualHeight;

            if (md_shape == null)
                return -1;
            else
                return md_shape.mvf_GetVolumeWithHeight(md_actualHeight);
        }

    }

    public class VolumeEventArgs : MeasurementEventArgs
    {
        public int md_volume;


        public VolumeEventArgs(double temperature , double humidity, int volume)
            : base(temperature, humidity)
        {
            md_volume = volume;
        }

    }
}
```

Επίσης υπάρχει και η αφηρημένη κλάση σχήμα :

```
using System;
using Microsoft.SPOT;

namespace TestRightView.Control
{
```

```
 // afirimeni clasi sxima
   public abstract  class Shape
   {

      public abstract int mvf_GetMaxVolume();
      public abstract int mvf_GetVolumeWithHeight(int height);
   }
}
```

Λόγο του ότι είναι δύσκολο να πάρουμε τιμές από έναν αισθητήρα πίεσης χρησιμοποιήθηκε ένας αισθητήρας θερμοκρασίας ο SHT15, ο οποίος τροφοδοτούσε με δεδομένα το πρόγραμμα. Ο οδηγός του αισθητήρα ήταν έτοιμος και απλά προστέθηκε στο όλο πρόγραμμα.

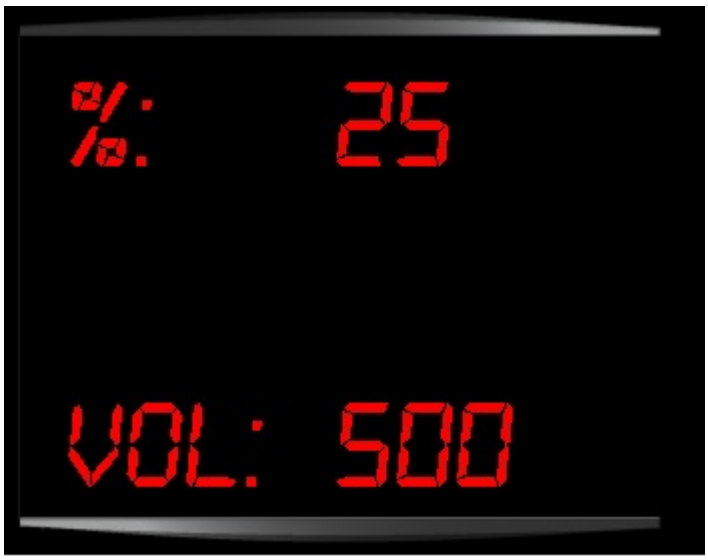
```
using System;
using System.Threading;
using Microsoft.SPOT;

using ElzeKool.io;
using ElzeKool.io.sht11_io;

namespace TestRightView.Control
{
   public class Sensor
   {
      public delegate void HaveMeasurementEventHander(Object sender,
MeasurementEventArgs e);
      public event HaveMeasurementEventHander HaveMeasurement;

      private Timer md_reportTimer;
      private SensirionSHT11 md_sht11;

      public Sensor(SensirionSHT11 sht11)
      {
         md_sht11 = sht11;

      }

      public bool mf_init()
      {

         bool initialized = false;

         // Soft-Reset the SHT11
         if (md_sht11.SoftReset())
         {
            initialized = false;
```

```
  // Softreset returns True on error
                //throw new Exception("Error while resetting SHT11");
                return initialized;
            }

            // Set Temperature and Humidity to less acurate 12/8 bit
            if
(md_sht11.WriteStatusRegister(SensirionSHT11.SHT11Settings.LessAcurate))
            {
                // WriteRegister returns True on error
                //throw new Exception("Error while writing status register SHT11");
                return initialized;
            }

            // Do readout
            Debug.Print("RAW Temperature 12-Bit: " +
md_sht11.ReadTemperatureRaw());
            Debug.Print("RAW Humidity 8-Bit: " + md_sht11.ReadHumidityRaw());

            // Set Temperature and Humidity to more acurate 14/12 bit
            if (md_sht11.WriteStatusRegister((SensirionSHT11.SHT11Settings.NullFlag)))
            {
                // WriteRegister returns True on error
                //throw new Exception("Error while writing status register SHT11");
                return initialized;
            }

            // Do readout
            Debug.Print("RAW Temperature 14-Bit: " +
md_sht11.ReadTemperatureRaw());
            Debug.Print("RAW Humidity 12-Bit: " + md_sht11.ReadHumidityRaw());

            initialized = true;

            return initialized;
        }

        public bool mf_StartReport()
        {
            TimeSpan delayTime = new TimeSpan(0, 0, 5);

            //orismos xronou deigmatolipsias  5 sec
            TimeSpan intervalTime = new TimeSpan(0, 0, 0, 5, 0);
            md_reportTimer = new Timer(new TimerCallback(ReportCallback), null,
delayTime, intervalTime);

            return true;
        }
```

```
  public double mf_ReadTemperature()
      {
          return
md_sht11.ReadTemperature(SensirionSHT11.SHT11VDD_Voltages.VDD_5V,
SensirionSHT11.SHT11TemperatureUnits.Celcius);
      }


      public double mf_ReadHumidity()
      {
          return
md_sht11.ReadRelativeHumidity(SensirionSHT11.SHT11VDD_Voltages.VDD_5V);
      }


      private void ReportCallback(object state)
      {
          double temp;
          double humidity;

          try
          {
              temp =mf_ReadTemperature(); //temp = 30;
              humidity = mf_ReadHumidity(); //humidity = 30;

              MeasurementEventArgs e = new MeasurementEventArgs(temp, humidity);
              mf_HaveMeasurement(e);
          }
          catch (Exception ex)
          {
              Debug.Print("SEnsor is dead");
          }




      }

      private void mf_HaveMeasurement(MeasurementEventArgs e)
      {
          if (HaveMeasurement != null)
              HaveMeasurement(this, e);
      }
  }


   public class MeasurementEventArgs : EventArgs
  {
```

```
  public double md_temperature;
      public double md_humidity;


      public MeasurementEventArgs(double temperature , double humidity)
         : base()
      {
         md_temperature = temperature;
         md_humidity = humidity;
      }

   }
}
```

Αφού γίνει η εισαγωγή των στοιχείων και πατήσουμε το OK το σύστημα μας εμφανίζει σε δεύτερη οθόνη τον όγκο σε λίτρα που έχει υπολογίσει με βάση τα στοιχεία που έχουμε δώσει και τα δεδομένα που λαμβάνει από τον αισθητήρα που στην συγκεκριμένη περίπτωση αναπαριστούν την πίεση που θα αντιλαμβανόταν ο αισθητήρας πίεσης από μια στήλη υγρού στην επιφάνεια μέτρησής του.



Η οθόνη αυτή υλοποιείτε από τον παρακάτω κώδικα :

```
using System;
using Microsoft.SPOT;
using Microsoft.SPOT.IO;
using System.Threading;
using Microsoft.SPOT.Hardware;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Controls;
using Microsoft.SPOT.Presentation.Media;
using Bluedev.NMF.Controls;



namespace TestRightView
{
   public class MeasureCanvas:Canvas
   {

      private StackPanel md_HumidityPanel;
      private gc_GUILabel md_HumidityTextBox;


      private StackPanel md_TempPanel;
      private gc_GUILabel md_TempTextBox;

      private StackPanel md_VolPanel;
      private gc_GUILabel md_VolTextBox;


      public MeasureCanvas()
         :base()
      {

         Font small = Resources.GetFont(Resources.FontResources.Digital_10);
         Font big = Resources.GetFont(Resources.FontResources.Digital_22);


            //TextBox width
            md_TempTextBox = new gc_GUILabel("0", big);
            md_TempPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_TempPanel, md_TempTextBox, "% ", 20, 20,
Visibility.Visible);

            //TextBox width
           /*  md_HumidityTextBox = new gc_GUILabel("0", big);
            md_HumidityPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_HumidityPanel, md_HumidityTextBox, "Hum", 90, 20,
Visibility.Visible);*/
```

```
//TextBox Volume
            md_VolTextBox = new gc_GUILabel("0", big);
            md_VolPanel = new StackPanel(Orientation.Horizontal);
            AddTextBox(md_VolPanel, md_VolTextBox, "Vol", 130, 20,
Visibility.Visible);

            Program.s_Sensor.mf_StartReport();
            Program.s_Tank.HaveMeasurement += new
TestRightView.Control.Tank.HaveMeasurementEventHander(s_Tank_HaveMeasure
ment);




        }

        void s_Tank_HaveMeasurement(object sender, Control.VolumeEventArgs e)
        {
            Program.s_measureCanvas.Dispatcher.BeginInvoke(new
DispatcherOperationCallback(mf_UpdateMeasurement), e);
        }


        private object mf_UpdateMeasurement(object arg)
        {
            Control.VolumeEventArgs measurement = (Control.VolumeEventArgs)arg;

            md_TempTextBox.mp_TextContent =
measurement.md_temperature.ToString("f0");
          //md_HumidityTextBox.mp_TextContent =
measurement.md_humidity.ToString("f2");
            md_VolTextBox.mp_TextContent = measurement.md_volume.ToString();

            return null;

        }




        private void AddTextBox(StackPanel panel, gc_GUILabel labelMeasure, String
caption, int top, int left,Visibility visibility)
        {

            Font big = Resources.GetFont(Resources.FontResources.Digital_22);
            Font small = Resources.GetFont(Resources.FontResources.Digital_10);
```

```
 //Label
        gc_GUILabel label = new gc_GUILabel(caption +":", big);
        label.mp_Background = null;
        label.mp_Border = null;
        label.mp_ForeColor = Colors.Red;
        label.Width = 135;
        panel.Children.Add(label);


        labelMeasure.mp_Background = null;
        labelMeasure.mp_Border = null;
        labelMeasure.mp_ForeColor = Colors.Red;
        labelMeasure.Width = 135;
        panel.Children.Add(labelMeasure);

        //Add to stackpanel
        Children.Add(panel);
        Canvas.SetTop(panel, top);
        Canvas.SetLeft(panel, left);

        panel.Visibility = visibility;


    }

  }

}
```

Ένα άλλο σημαντικό στοιχείο στο πρόγραμμα το οποίο είναι απαραίτητο για να μπορούμε ανά πάσα στιγμή να ελέγξουμε πόσα λίτρα υπήρχαν στην δεξαμενή, σε πόσο καιρό καταναλώθηκαν , ποιες ώρες γινόταν κατανάλωση , πότε προσθέσαμε πετρέλαιο και πόσα λίτρα, όταν γίνετε κατανάλωση πόσα λίτρα καίμε σε π.χ 30min , είναι το αρχείο καταγραφής που βρίσκετε στην SD κάρτα μνήμης. Με μια κάρτα 1GB και με δειγματοληψία κάθε 5sec μπορούμε να έχουμε ένα αρχείο άνω των 6 ετών.

Η αποθήκευση τις ημερομηνίας της ώρας και της ποσότητας του καυσίμου στη κάρτα μνήμης γίνετε από τον παρακάτω κώδικα:

```
using System;
using System.IO;
using System.Text;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.IO;
using Microsoft.SPOT.Hardware;
```

```
using TestRightView.Control;


namespace TestRightView.Storage
{
   public class SDLoger
   {
      private enum SDCardState
      {
         EJECTED,
         INSERTED,
         ERROR
      }

      private SDCardState _sdCardState;
      private FileStream _fileHandle;

      public SDLoger()
      {

         //Search for an inserted sd card
         VolumeInfo[] volumeInfos = VolumeInfo.GetVolumes();
         if (volumeInfos.Length != 0)
         {
            _sdCardState = SDCardState.INSERTED;
            string rootDirectory = VolumeInfo.GetVolumes()[0].RootDirectory;
            _fileHandle = new FileStream(rootDirectory + @"\tank_data.txt",
FileMode.OpenOrCreate);
            _fileHandle.Seek(_fileHandle.Length, 0);
         }
         else
            _sdCardState = SDCardState.EJECTED;

         RemovableMedia.Insert += new
InsertEventHandler(RemovableMedia_Insert);
         RemovableMedia.Eject += new EjectEventHandler(RemovableMedia_Eject);

      // set the date and time     year  month day  hour minute  second  millisecond
         DateTime time = new DateTime(2011, 9,    16,  22,   44,     50,     0);
         Utility.SetLocalTime(time);

         Debug.Print(DateTime.Now.ToString());
      }



      void RemovableMedia_Eject(object sender, MediaEventArgs e)
      {
         _sdCardState = SDCardState.EJECTED;
```

```
            if (_fileHandle != null)
            {
                _fileHandle.Close();
                _fileHandle.Dispose();
                _fileHandle = null;
            }

        }

        void RemovableMedia_Insert(object sender, MediaEventArgs e)
        {
            _sdCardState = SDCardState.INSERTED;

            if (_fileHandle == null)
            {
                string rootDirectory = VolumeInfo.GetVolumes()[0].RootDirectory;
                _fileHandle = new FileStream(rootDirectory + @"\tank_data.txt",
FileMode.OpenOrCreate);
                _fileHandle.Seek(_fileHandle.Length, 0);
            }


        }

    public void mf_WriteEntry(string str)
    {

      try
        {
          byte[] data = Encoding.UTF8.GetBytes(DateTime.Now.ToString() + " : " + str
+" "+ "\n" );
             // byte[] data1 = Encoding.UTF8.GetBytes("  "+"\n");
              //_fileHandle.Write(data1,0,data1.Length);
                _fileHandle.Write(data, 0, data.Length);

                _fileHandle.Flush();
            }
            catch (Exception e)
            {
                Debug.Print(e.ToString());

                _fileHandle.Close();
                _fileHandle.Dispose();
                _fileHandle = null;
            }


        }
```

```
    public void mf_HaveMeasurmentEventHandler(object sender, VolumeEventArgs e)
        {
            if (_sdCardState == SDCardState.INSERTED )


                mf_WriteEntry(e.md_volume.ToString());

        }

    }
}
```

Ένα μικρο δείγμα τις καταγραφής ακολουθεί παρακάτω.

```
09/16/2011 23:03:00 : 320
09/16/2011 23:03:05 : 320
09/16/2011 23:03:10 : 320
09/16/2011 23:03:15 : 320
09/16/2011 23:03:20 : 320
09/16/2011 23:03:25 : 320
09/16/2011 23:03:30 : 320
09/16/2011 23:03:35 : 310
09/16/2011 23:03:40 : 310
09/16/2011 23:03:45 : 300
09/16/2011 23:03:50 : 300
09/16/2011 23:03:55 : 300
09/16/2011 23:04:00 : 290
09/16/2011 23:04:05 : 290
09/16/2011 23:04:10 : 290
```

Τέλος έχουμε το τελευταίο τμήμα του προγράμματος όπου περιλαμβάνετε η main .

```
using System;
using System.IO;
using System.Threading;
using System.Reflection;

using Microsoft.SPOT;
using Microsoft.SPOT.IO;
using Microsoft.SPOT.Input;
using Microsoft.SPOT.Hardware;
```

```
using Microsoft.SPOT.Presentation;

using Microsoft.SPOT.Presentation.Media;
using Microsoft.SPOT.Presentation.Controls;

using TestRightView.Control;
using TestRightView.Storage;

using ElzeKool.io;
using ElzeKool.io.sht11_io;

namespace TestRightView
{
   public class Program : Microsoft.SPOT.Application
   {
      public static Tank s_Tank;
      public static Sensor s_Sensor;
      public static SDLoger s_sdManager;


      public static MeasureCanvas s_measureCanvas;

      public static void Main()
      {
         Program myApplication = new Program();

         Microsoft.SPOT.Touch.Touch.Initialize(myApplication);

         Window mainWindow = myApplication.CreateWindow();

         // Create the object that configures the GPIO pins to buttons.
         GPIOButtonInputProvider inputProvider = new
GPIOButtonInputProvider(null);


         //Initialize sensor
         SHT11_GPIO_IOProvider SHT11_IO = new
SHT11_GPIO_IOProvider((Cpu.Pin)15, (Cpu.Pin)16);
         s_Sensor = new Sensor(new SensirionSHT11(SHT11_IO));

         if (!s_Sensor.mf_init())
         {
            throw new Exception("Error initializing SHT11");
         }

         s_Tank = new Tank();

         s_sdManager = new SDLoger();
```

```
 s_Tank.HaveMeasurement += new
Tank.HaveMeasurementEventHander(s_sdManager.mf_HaveMeasurmentEventHan
dler);


            // Start the application
            myApplication.Run(mainWindow);
        }

        private Window mainWindow;

        public Window CreateWindow()
        {
            // Create a window object and set its size to the
            // size of the display.
            mainWindow = new Window();
            mainWindow.Height = SystemMetrics.ScreenHeight;
            mainWindow.Width = SystemMetrics.ScreenWidth;

            // Create a single text control.
            Text text = new Text();

            text.Font = Resources.GetFont(Resources.FontResources.small);
            text.TextContent =
Resources.GetString(Resources.StringResources.String1);
            text.HorizontalAlignment =
Microsoft.SPOT.Presentation.HorizontalAlignment.Center;
            text.VerticalAlignment =
Microsoft.SPOT.Presentation.VerticalAlignment.Center;

            // Add the text control to the window.


            OptionsPanel optionCanvas = new OptionsPanel();
            optionCanvas.Finished += ChangetoMeasureCanvas;
            mainWindow.Child = optionCanvas;


            // Set the window visibility to visible.
            mainWindow.Visibility = Visibility.Visible;

            mainWindow.Background = new
SolidColorBrush(Colors.Black/*ColorUtility.ColorFromRGB(0x99, 0x99, 0xFF)*/);

            // Attach the button focus to the window.
            Buttons.Focus(mainWindow);

            return mainWindow;
        }
```

```
        private void ChangetoMeasureCanvas(Object sender, OptionsEventArgs e)
        {
            Type type = e.md_info.GetType();

            if(type.Name == "CubeInfo")
            {
                CubeInfo cubeinfo = (CubeInfo)e.md_info;
                s_Tank.mf_SetShape(new Cube(cubeinfo.md_height, cubeinfo.md_width,
cubeinfo.md_lenght));
            }
            else
            {
                CylinderInfo cylinderinfo = (CylinderInfo)e.md_info;
                s_Tank.mf_SetShape(new Cylinder(cylinderinfo.md_height,
cylinderinfo.md_radius));
            }


            Debug.Print(s_Tank.mf_GetMaxVolume().ToString());


            s_measureCanvas = new MeasureCanvas();
            mainWindow.Child = s_measureCanvas;
        }


    }
}
```

Για την λειτουργία του προγράμματος χρησιμοποιήθηκαν και άλλες έτοιμες βιβλιοθήκες για το .NET Micro Framework που έχουν φτιαχτεί από την εταιρία BLUEdev η οποία αναπτύσσει λογισμικό για embedded συστήματα, οι οποίες δεν υπάρχει λόγος να εξηγηθούν αναλυτικά στην συγκεκριμένη εργασία.

10. Συμπεράσματα

Η τεχνολογία στις μέρες μας προσφέρει πάρα πολλές λύσεις που καλύπτουν οποιασδήποτε ανάγκη. Μεγάλες εταιρίες προσφέρουν έτοιμα συστήματα τα οποία συνήθως απευθύνονται σε μεγάλες κτιριακές .

Ένας από τους στόχους τις παραπάνω εργασίας είναι να προσφέρει μια οικονομική και αξιόπιστη λύση για τις είδη υπάρχουσες εγκαταστάσεις θέρμανσης χωρίς ιδιαίτερες απαιτήσεις και μετατροπές. Χρησιμοποιώντας το παραπάνω σύστημα δίνει στον χρήστη την δυνατότητα να έχει ανά πάσα στιγμή όλα τα στοιχεία τις δεξαμενής , ποια είναι η υπάρχουσα ποσότητα κτλ. Η καταγραφή γίνετε σε εξωτερική μνήμη έτσι τα δεδομένα μπορούν να μεταφερθούν σε οποιοδήποτε υπολογιστή για περαιτέρω επεξεργασία.

Μια άλλη προσέγγιση που θα μπορούσε να είχε υλοποιηθεί είναι η αποστολή των δεδομένων στο internet όμως αυτό απαιτεί πρόσβαση στο internet και απαραίτητος είναι ένας server που θα αποθηκεύονται όλα τα δεδομένα, γι' αυτό προτιμήθηκε η λυση τις εξωτερικής μνήμης.

Λύσεις υπάρχουν σχεδόν για όλες τις εφαρμογές αυτό που είναι σημαντικό είναι να εφαρμόζετε η σωστή λύση για την εκάστοτε εφαρμογή, έτσι και η παραπάνω πτυχιακή εργασία δίνει μια ακόμα λύση στην διαχείριση του αποθέματος πετρελαίου θέρμανσης κυρίως σε οικιακές εγκαταστάσεις θέρμανσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) Expert .NET Micro Framework (2nd ed.Sep 2009)
2) Beginners guide to NETMF
3) csharp_ebook
4) Learning UML 2.0-K.Hamilton, R.Miles- O"Reilly- 2006
5) anaptyksh_efarmogwn_me_.net_dotnet_microsoft_1
6) UML_lecture1
7) UML.Distilled.(3ed.2004).-Addison.Wesley
8) .net micro framework white paper
9) oop_lecture09_introtoc_and_4site